TRANSACT 4

WSJ Framework API

Programmer's Guide

9/22/99

Change bars indicate updates from 7/6/99.

OPEN MARKET

We ARE Internet Commerce.

This product contains encryption technology from RSA Data Security, Inc.

Open Market, Inc.
One Wayside Road, Burlington, MA 01803
T: 781.359.3000
www.openmarket.com

Table of

Contents

CHAPTER 1

Introduction

This manual explains how to use the API provided by the WSJIE (Wall Street Journal Interactive Edition) custom classes.

The API is designed to support group access control, external subsystem integration, and customer service customizations. The system uses Sun SPARC hardware and the SOLARIS (Unix) operating system.

This chapter answers a few common questions about the API.

How Do I Read this Manual in PDF?

When you use the Adobe Acrobat Reader to read the PDF version of this document, you might wish to use these features:

- A bookmark column showing chapters, classes, and methods in a tree stucture. (Use View > Bookmarks and Page to display the column; click the section name to open, or click the triangles to open or close a branch.)
- "Live" pointers on the Table of Contents and Index let you click to section title or page number to display the associated page. (Use View > Go To Page 3 to display the Table of Contents.)
- Text search (use Tools > Find).
- Text copy (use Tools > Select Text to change the cursor).

What Does a WSJ Framework Application Do?

The custom applications that you can write handle activities related to providing WSJIE subscriptions and group access to buyers.

The applications must run on Transact front-host machines. This is where the applications that access the database are usually installed.

How Does the API Relate to Transact?

The WSJ Customizations, including the API, interact with Transact as shown in the following figure.

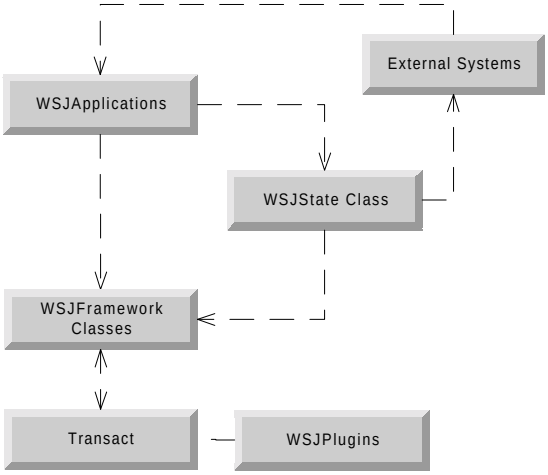

Figure 1: WSJ Customizations System Architecture

The modules shown in Figure 1 are as follows:

- WSJApplications - Custom buyer and admin applications, including the GUI applications for both buyer and admin (e.g., Registration, Your Account, Resource, Product, Offer, and Group CGI scripts); the "bulk buyer" applications `wsjModGroupApp` and `wsjRegSubscriberApp`, which modify group and individual subscriptions per specifications in flat files; and the "record injection applications that read the external OLF and NR records and perform the necessary C/SR level functions and buyer state transitions, if necessary.
- External Systems - Legacy components WSJ NR and OLF billing systems, and PrintOffer system.
- WSJState classes - A custom class to communicate to the external systems
- WSJFramework classes - Low-level custom classes to provide additional, custom Transact functionality and an API to which seller applications may be written.
- WSJPlugins - Additional required functionality not provided by WSJFramework classes.
- Transact - The standard Open Market transaction system.

Can I Modify Existing Transact Modules?

For security reasons, we do not ship the code for the out-of-the-box modules provided by Transact. Therefore, you cannot change their features.

What Software Do I Need to Develop a Custom Application?

To develop a custom application, you need:

- The Transact 4 Utilities subsystem, comprising header and library files in the `tms-ts/sdk/include` and `tms-ts/lib` directories. The utilities subsystem contains many generally useful classes. For more information, see *Transact 4 Utility Classes Programmers Guide.*
- The WSJ Framework API. The appropriate header and library files are also found in the `tms-ts/sdk/include` and `tms-ts/lib` directories.
- A C++ compiler that generates code for the operating system on which Transact 4 is running. The Solaris Sun WorkShop Compiler C++, version 4.2, is supported. Other compilers may work, but they are not supported. The GNU/g++ compiler is not supported and will not work.
- The dbx debugger.
- A web browser to test your application.

You also need access to a Unix system running Transact 4.

How Do I Build My Application?

This section explains how to install, create, compile, and test a WSJ Framework application.

The simplest way to compile such an application is with the Makefile included with the example programs. After you configure it, as explained below, you can copy it to the directory where your programs are, and edit it to fit your application.

An unconfigured demo Makefile is located in `tms-ts/sdk/demo/wsjframework/Makefile.in`. Other `Makefile.in` files are in other `tms-ts/sdk` subdirectories. You can use the `configure` script to process these files and produce a `Makefile` in each of the subdirectories where a `Makefile.in` is found. To use the script, issue the following command from within the `tms-ts/sdk` directory:

```
./configure --enable-optimization=on --enable-debug=off
```

If you wish to write your own Makefile from scratch, the information below will be essential.

Environment Variables

The following environment variables are important when compiling and using any Transact application:

`OMKT_REGISTRY_FILE`
: This variable is important at runtime. It must be set to indicate the *absolute* pathname of the Transact registry file. This file is usually contained in `tms-ts/conf`. It should be named `registry.asc`.

`LD_LIBRARY_PATH`
: This variable is used by the Solaris compiler to find the libraries used to link the compiled program. Make sure that the directory `tms-ts/lib` is on `LD_LIBRARY_PATH`.

`ORACLE_HOME`
: This variable references the Oracle installation directory.

Libraries

The WSJ Framework API requires most of the `.so` and `.api` files found in `tms-ts/lib`. It is important that those files be available for your application to run.

Header Files to Include

Your application's header file must itself include the following header files:

- WSJFoundation Factory.h

```
#ifndef _WSJ_FOUNDATION_FACTORY_H
#include <WSJFoundationFactory.h>
#endif
```

The WSJFoundationFactory.h header lets you create an instance of any foundation abstract base classes; thus it also includes all headers for those objects.

- WSJBuyerStateHandler.h

```
#ifndef _WSJ_BUYER_STATE_SERVICE_H
#include <WSJBuyerStateHandler.h>
#endif
```

The WSJBuyerStateHandler.h header lets you modifythe buyer state.

Debugging Your Application

To debug your application, you need to:

- Compile and link with the -debug flag.
- Use dbx as the debugger. (But don't expect to step into the libraries provided by Open Market.)

Examples

Demo application programs are provided with the WSJ Framework API. Each program exercises a particular set of functionality.

The demo programs are named and located as follows:

```
tms-ts/sdk/demo/wsjframework/CompBuyerTest.cpp
tms-ts/sdk/demo/wsjframework/FreeBuyerTest.cpp
tms-ts/sdk/demo/wsjframework/PaidBuyerTest.cpp
tms-ts/sdk/demo/wsjframework/AddProductTest.cpp
tms-ts/sdk/demo/wsjframework/AddResourceTest.cpp
tms-ts/sdk/demo/wsjframework/CompBuyerTest.cpp
tms-ts/sdk/demo/wsjframework/CreateOfferTest.cpp
tms-ts/sdk/demo/wsjframework/CreateProductTest.cpp
tms-ts/sdk/demo/wsjframework/CreateResourceTest.cpp
```

CHAPTER 2

Overview of WSJIE Classes

This chapter introduces class-related concepts and the two sets of classes that make up the custom application framework for extending the Open Market Transact Seller Application subsystem:

- Foundation classes
- State machine classes

Introductory Concepts and Terminology

The WSJ Framework API is primarily designed to support products and offers on Wall Street Journal web pages, particularly related to subscription information and allowing a buyer (subscriber) to register, subscribe, and (if not a group subscriber) pay, and a customer service representative (CSR) to administer the accounts and products, etc.

The terms *buyer* and *subscriber* are used more or less interchangeably.

The terms *periodical*, *resource*, and *domain* are often used interchangeably to mean the item to which a buyer will have or has subscribed.

The terms *product* and *offer* refer, respectively, to:

- Content (*resource*) that the subscriber can obtain — of a particular type based upon the payment plan (e.g., IPAS, INAS)
- The set of products offered to a subscriber

Note: Throughout this guide, object instances are commonly referred to without using the entire object name. Therefore, the following common terms refer to the objects shown below:

A *buyer* or *subscriber* is an instance of a VWSJBuyer.
A *domain* is an instance of a VWSJDomain.
An *offer* is an instance of a VWSJOffer.
A *product* is an instance of a VWSJProduct.
A *subscription* is an instance of a VWSJSubscription.

High-Level Entity Relation Diagram

The following figure illustrates how the more common classes are related:

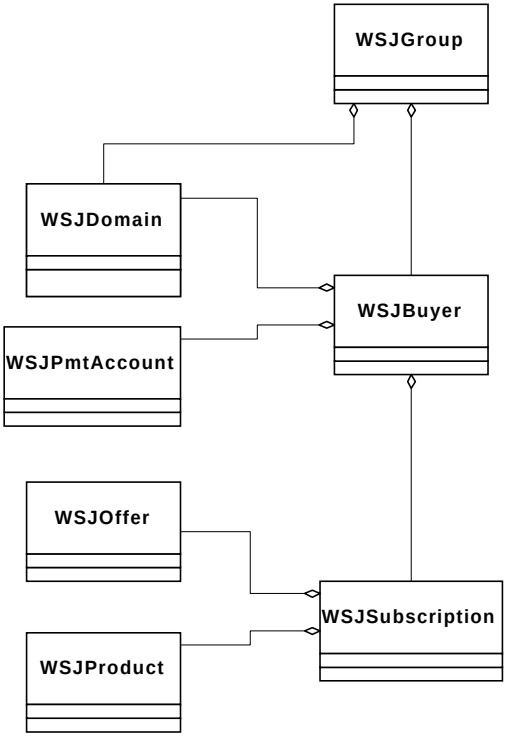

Figure 2: Entity Relation Diagram

Virtual and Template Classes

The WSJ Framework API presents two different sets of classes, identifiable by the first letter of their names:

- Virtual — The names of abstract base classes start with the letter "V." The constructors for these classes are not available for public use. (See below.)
- Template — The names of concrete classes start with the letter "T."

The WSJ Framework API has a class, TWSJFoundationFactory, that must be used to create all the VWSJ classes. Thus, the client is restricted from using the constructors for any of those classes. To create an instance of a VWSJ class, use:

```
TWSJFoundationFactory factory;
VWSJClassName *ClassName = factory.MakeVWSJClassName();
```

For example:

```
TWSJFoundationFactory factory;
VWSJBuyer *buyer = factory.MakeVWSJBuyer();
```

Paper/Service Classes

Methods in the TWSJFoundationFactory class create foundation objects. These in turn support the WSJ interfaces to the following objects (each is described later):

- Buyer and Buyer-related objects
- Context objects
- Subscription, Offer, Product, Group, and Domain objects
- Their respective Service objects

You can think of these foundation objects as "paper" objects in the sense that they can be created to receive data obtained, via WSJ Framework Service class methods, from a Transact database. As such, they're like pages in a printed book.

They are also "paper" objects in the sense that they can be created, written upon by your application program, and then have their contents written to the Transact database by other appropriate WSJ Framework Service class methods.

The following figure illustrates how paper classes, the FoundationFactory, and the Transact database interact:

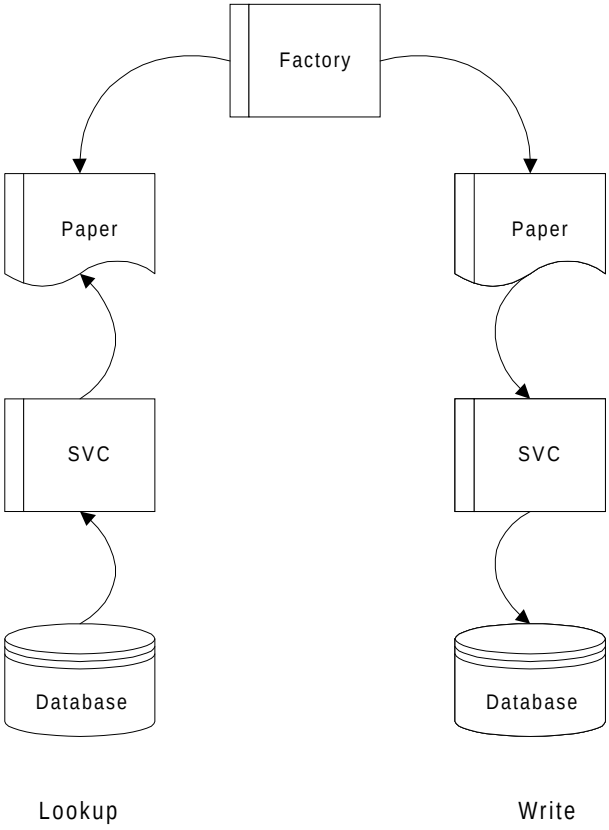

Figure 3: Paper Classes and the FoundationFactory

Application Context

The VWSJContext class provides a handle (an opaque pointer) to the global information needed by the API. This handle is passed by methods in the various classes as an input pointer to a VWSJContext constant. This pointer is used to specify the object containing the operational context required to perform database operations.

Database Concepts

Commits are hidden under the API and operate automatically.

Foundation Classes

These classes provide all the basic functionality necessary for performing WSJ-specific operations such as:

- Arbitrary IFP and expiration date extensions
- Configurable auto-renew
- History record generation and retrieval
- Other global data storage and access, including buyer status
- Payment account addition (in addition to standard Transact functionality, must generate 'new customer' NR record and DJ-account number)
- Print account number validation
- DJ-account number generation using an algorithm provided by WSJ but configurable by WSJ via a callout API
- Resource, offer, and product management

Paper Classes

Paper classes do not read or write to the database. Only service classes (and the state class) update and write to the Transact database itself.

Administration Classes

These are the primary classes that you will use:

- VWSJDomain — Methods in this class access and change properties such as a domain's name, ID, and URL.
- VWSJGroup — Methods in this class access and change properties such as a subscriber's group's name, description, address, phone, and maximum number of subscribers allowed.
- VWSJOffer — Methods in this class access and change properties related to offers being made to a buyer. These include the offer ID, name, description, number of days, and miscellaneous data; as well as other agreement registration, filename, date last modified, and creator ID.

Buyer Classes

- VWSJBuyer — Methods in this class access and change properties such as a buyer's ID, account number, password, name, and subscription-related data.

- VWSJPmtAccount — Methods in this class access and change properties related to a buyer's credit card, e.g., expiration date, brand, name on card, number, and type.
- VWSJProduct — Methods in this class add, remove, access, and change properties related to the product that a buyer is trying to buy. These include the product's name, cost, and related resources; whether it is available, etc.
- VWSJSubscription — Methods in this class center on starting, ending, and getting a subscription, along with accessing and setting product, offer, and subscriber IDs.

Service Classes

Service classes update and write to the Transact database itself using the "paper" object as a parameter. (Other classes' methods operate on class properties.)

For Administration Objects

These classes cause reads and write to the Transact database, as well as automatic commits:

- VWSJDomainService — Methods in this class create domains; look them up by ID or name; update them; and get their names.
- VWSJGroupService — Methods in this class create groups; look them up by ID or name; update them; and get their names.
- VWSJOfferService — Methods in this class create offers; look them up by ID or name; update them; and add products to them.
- VWSJProductService — Methods in this class create products and look them all up, or look up individual products by ID or name.

For Buyer Objects

Service classes for buyer objects only read from the Transact database. You must use methods in the TWSJBuyerStateHandler class to write to the database or propagate the writes to any external systems.

The buyer service classes include:

- VWSJBuyerService — Methods in this class look up or change buyer-related information such as password, account secret, mail alerts, according to buyer by name, ID, or DJ Account number.

- VWSJSubscriptionService — Methods in this class look up subscription information according to buyer or group ID; suspend and reinstate subscribers; cancel groups and individuals; specify how much to charge for a subscription; and specify that a subscriber can not renew.
- VWSJPmtAccountService — Methods in this class look up payment accounts by ID.

Collaborative Classes

These classes are used by the VWSJBuyer classes listed above:

- TWSJProductHistory and TWSJProductHistoryService — Methods in these classes access and change properties related to a product ID.
- TWSJSearchCriteria — Methods in these classes access and change properties related to a subscriber's account data, such as address, activity, cost, email, names, etc.
- TWSJSuspensionHistory and TWSJSuspensionHistoryService — Methods in these classes access and change properties related to a buyer's suspension history: subscriber and periodical ID; status, suspension date and history, comments, etc.

Vector Classes

Vector classes implement growable arrays of pointers to objects that can be accessed by an integer index. A vector can grow or shrink to add or remove items (after it has been created).

The vector classes all have the same interface, described under the VWSJ*Object*Vector classes in the next chapter.

- VWSJAccessVector (this class works with AccessHistory objecs)
- VWSJDomainVector
- VWSJExtensionVector (this class works with ExtensionHistory objects)
- VWSJGroupVector
- VWSJPmtAccountVector (this class works with PmtAccountHistory objects)
- VWSJProductHistoryVector
- VWSJProductVector
- VWSJSuspensionVector (this class works with SuspensionHistory objects)

State Class

This class is responsible for state transition mappings. This has the advantage of separating the state diagram from the rest of the system to allow maximum flexibility should new states or transitions be required.

The only state class included in the WSJ Framework API is:

- TWSJBuyerStateHandler

CHAPTER 3

The WSJ Framework API

Chapter 2 explained how to code an example application. This chapter, Chapter 3, lists the WSJ Framework API classes alphabetically, along with their public methods and data elements. Any higher-level applications should write to this framework API, because certain function calls cause interactions with external components unsupported by the analogous core Transact calls.

Common Methods

Many of the WSJ Framework API classes support the same `IsValid` method. This method checks the validity of the member fields and returns `TRUE or FALSE.`

```
virtual Bool IsValid () const = 0;
```

Other API classes support the same `Test` method:

```
static Bool Test();
```

Common Inputs

For *all* `Set`*`Data`* methods, the following input is used:

`Value`
: The data item being set.

Common Return Values

Most methods return the value you are requesting, as well as the following in HRESULT:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

TWSJ*Object*Vector Class

Name

`TWSJ`*`Object`*`Vector` – Classes for managing vectors of objects.

Synopsis (Constructors and Destructor)

```
TWSJObjectVector () {};                    // common constructor

TWSJDomainVector (const int length) {};    // additional constructor
TWSJGroupVector (const int length) {};     // additionl constructor
TWSJProductVector (const int length) {};   // additional constructor

TWSJObjectVector (const TWSJObjectVector& rhs)) {};// copy constructor
virtual ~TWSJObjectVector () {};       // destructor
```

Description

The following TWSJ foundation objects support a vector class: Access, Domain, Extension, Group, Product History, and Suspension

Methods in these objects' associated vector classes look up, append, measure, and clear a vector or sequence of TWSJ objects that can be iterated over.

Public Operators

```
Bool operator==(const TWSJObjectVector& rh) const;// equality
Bool operator!=(const TWSJObjectVector& rh) const;// inequality
TWSJObjectVector& operator=(
                        const TWSJObjectVector& rh); // assignment

VWSJDomain*& operator[](const int& index);          // [] operator
```

Public Methods

All the TWSJ*Object*Vector classes support the following methods:

- `static Bool Test();`
- Those listed in Table 1, "Vector Methods for TWSJObjectVector Classes," below.

Table 1: Vector Methods for TWSJ*Object*Vector Classes

Vector Method	Access *	Domain	Extension *	Group	Pmt Account *	Product	Product History	Suspension *
`Append`	`yes`	`no`	yes	no	yes	no	yes	yes
`Clear`	`yes`	`yes`	yes	yes	yes	yes	yes	yes
`GetAt`	`yes`	`yes`	yes	yes	yes	yes	yes	yes
`Length`	`yes`	`yes`	yes	yes	yes	yes	yes	yes
`SetAt`	`no`	`yes`	no	yes	no	no	no	no

* For the TWSJAccessVector, TWSJExtensionVector, TWSJPmtAccountVector, and TWSJSuspensionVector classes, the objects are TWSJAccessHistory, TWSJExtensionHistory, TWSJPmtAccountHistory, and TWSJSuspensionHistory objects, respectively.

Example

```
TWSJAccessVector vec;
...
for (int i = 0; i < vec.Length(); ++i) {
    TWSJAccessHistory val;
    Bool res = vec.GetAt(i, val);
    ASSERT(res);
    ...
}
```

TWSJ*Object*Vector::Append

Name

`Append` – Append a TWSJ object to the vector.

Synopsis

```
void Append(
              TWSJObject& val);
```

Description

This method appends an object to the vector.

It is available for the following classes: TWSJAccessVector, ExtensionVector, PmtAccountVector, ProductHistoryVector, and SuspensionVector.

Parameters

`val`

Input. The object to append to the vector. For the classes supporting this method, the objects in the vector are TWSJAccessHistory, ExtensionHistory, PmtAccountHistory, and SuspensionHistory objects.

Return Values

None.

Example

```
TGUIDStringVector resourceIDs;
resourceIDs.Append("25cb4174dd6b11d28101ac5fd6f63f05");
resourceIDs.Append("528486eedd6b11d28b66d99df3840ffa");
```

TWSJ*Object*Vector::Clear

Name

`Clear` – Clear a vector of all objects

Synopsis

```
void Clear();
```

Description

This method clears all objects from the vector.

It is available in all classes that support vectors: TWSJAccessVector, DomainVector, ExtensionVector, GroupVector, PmtAccountVector, ProductHistoryVector, and SuspensionVector.

Parameters

None

Return Values

None.

TWSJ*Object*Vector::GetAt

Name

`GetAt` – Get an object from the vector.

Synopsis

```
Bool GetAt(
                  const int& index,
                  TWSJObject& val);
```

Description

This method gets an object from the vector according to the specified index

It is available in all classes that support vectors: TWSJAccessVector, DomainVector, ExtensionVector, GroupVector, PmtAccountVector, ProductHistoryVector, and SuspensionVector.

Parameters

`index`

Input. An index value pointing to the object to retrieve from the vector. Index values are zero-based.

`val`

Output. The object retrieved from the vector. For TWSJAccessVector, ExtensionVector, PmtAccountVector, and ProductHistoryVector, the objects in the vector are TWSJAccessHistory, ExtensionHistory, PmtAccountHistory, ProductHistory, and SuspensionHistory objects.

Return Values

The `GetAt` method returns:

`TRUE`

If the method was successful.

`FALSE`

If the index is out of range.

TWSJ*Object*Vector::Length

Name

`Length` – Return the length of the vector.

Synopsis

```
int Length() const;
```

Description

This method returns the length of the vector.

It is available in all classes that support vectors: TWSJAccessVector, DomainVector, ExtensionVector, GroupVector, PmtAccountVector, ProductVector, and SuspensionVector.

Parameters

None

Return Values

The `Length` method returns the length of the vector as an integer.

Example

```
TWSJDomainVector domainVector;
domainService->GetAllDomains (wsjCtx, domainVector);

for (int i=0; i<domainVector.Length(); i++) {
    smartData.curResourcesLabels.insert (domainVector[i]->GetName ());
     smartData.curResourcesValues.insert (domainVector[i]->GetID());
}
```

TWSJ*Object*Vector::SetAt

Name

`SetAt` – Set an object into the vector at the given index.

Synopsis

```
Bool SetAt(
                  const int& index,
                  VWSJObject* const  & val);
```

Description

This method sets an object into the vector at the specified index.

It is available in the classes TWSJDomainVector and TWSJGroupVector.

Parameters

`index`
: Input. An index value pointing to the location in the vector to set. Index values are zero-based.

`val`
: Intput. The object to set into the vector.

Return Values

The `SetAt` method returns:

`TRUE`
: If the method was successful.

`FALSE`
: If the index is out of range.

TWSJBuyerStateHandler Class

Name

`TWSJBuyerStateHandler` – A WSJ buyer state class.

Synopsis (Constructor and Destructor)

```
TWSJBuyerStateHandler (VWMSJContext *ctx, *Buyer) {};
TWSJBuyerStateHandler () {};// destructor
```

Description

Methods in this class support the purging, activation, freezing, canceling, and updating of buyers.

Public Methods

```
HRESULT Activate();
HRESULT Cancel();
HRESULT Freeze();
HRESULT Purge();
HRESULT UpdateInfo();

Bool IsValid() const;
```

Related Classes

`VWSJSubscriptionService`

TWSJBuyerStateHandler::*Methods*

Name

`Activate, Cancel, Freeze, Purge, UpdateInfo` – Update the buyer's state.

Synopsis

```
HRESULT Activate();
HRESULT Cancel();
HRESULT Freeze();
HRESULT Purge();
HRESULT UpdateInfo();
```

Description

Methods in this class support the purging, activation, freezing, canceling, and updating of buyers, thus affecting their status. (See `GetSubscriberStatus` and `SetSubscriberStatus` methods for the VWSJBuyer class.)

All internal WSJIE transactions will be performed prior to creating records for external subsystems.

If there is an error in any of the internal transactions, an error will be logged, WSJIE will attempt to complete as much of the internal processing as possible, but no external systems will be processed.

If there is an error in any of the external systems, the error will be logged, and processing will continue to completion on all other registered external subsystems.

Activate

Use this method to:

- Do the initial activation (registration) of a subscriber (buyer) into the system
- Activate (re-instate) a suspended, frozen, or canceled subscriber
- Activate (convert) a canceled paid subscriber to a comp (complimenetary)
- Activate (convert) an expired comp subscriber to a paid buyer
- Activate a non-active (non-subscribed) buyer by adding a CC

Cancel

Cancel any eligible active buyer

Freeze

Freeze a free or an active paid subscriber, or suspend an active comp subscriber. If called for a buyer in IFP, no processing occurs and S_FALSE is returned.

Purge

Currently, this is only called in response to an OLF C record, or if a buyer registers but does not accept the subscriber agreement. Buyer must be in a frozen or suspended state, or else no processing will occur and an S_FALSE will be returned.

UpdateInfo

This is used to update any of the following information:

- Modify existing CC info, or add an additional CC account (to create the first CC account implies activation of a paid buyer, thus `Activate()` must be called).
- Any buyer info, e.g., Company, last name, first name, address, city, state, zip, country, phone number, email address, print account number, maiden name.
- The do-not-renew option.
- Pay plan info for active buyers; non-active buyers require `Activate()` to both activate and add a new pay plan. This information includes extensions, extension deletions, and replacing current payment plan (offer, product).

Return Values

All the buyer state handler methods return:

S_FALSE
: If the method was called for a buyer in an incompatible state, e.g., "activate" on an active buyer. No processing will occur.

E_FALSE
: If the method was not successful.

TWSJExtensionHistory Class

Name

`TWSJExtensionHistory` – The WSJ extension history class.

Synopsis (Constructor and Destructor)

```
TWSJExtensionHistory () {};                 // constructor
TWSJExtensionHistory (
              const TWSJExtensionHistory&)) {}; // copy constructor
virtual ~TWSJExtensionHistory() {};         // destructor
```

Description

Methods in this class create and change properties related to extensions (including the Initial Free Period) to a subscription.

Operators

```
Bool operator==(const TWSJExtensionHistory&) const;// equality
Bool operator!=(const TWSJExtensionHistory&) const;// inequality
TWSJExtensionHistory& operator=(
                        const TWSJExtensionHistory&); // assignment
friend ostream& operator<<(
                 ostream& ost,
                 const TWSJExtensionHistory& t);// insertion
```

Public Methods

Get*Data*
Set*Data*

```
Bool IsValid() const;
```

Related Classes

`TWSJExtensionHistoryService`

TWSJExtensionHistory::Get*Data*

Name

`Get`*`Data`* – Access extension history data.

Synopsis

```
virtual type GetValue () const;
```

Description

This group of methods accesses data values in the TWSJExtensionHistory class.

Parameters

None

Table 2: Accessor Methods for TWSJExtensionHistory Class

Return Type	Method	Description and Notes
`TDate`	`GetCreateTime`	When a particular extension was created
`TDate`	`GetExtensionDays`	Days in the extension (in seconds)
`GUIDString`	`GetExtensionHistoryID`	GUID for this extension history
`TString`	`GetExtensionType`	EXT (expiration); IFP (Initial Free Period); UNK (unknown)
`GUIDString`	`GetProductID`	GUID for the product belonging to buyer
`GUIDString`	`GetSubscriberID`	GUID for the buyer
`GUIDString`	`GetSubscriptionID`	GUID for the subscription

Notes

An *extension* is an addition to the number of days in a subscription. The *product* is an offering. A *subscription* is an instance of the product.

TWSJExtensionHistory::Set*Data*

Name

`Set*Data*` – Specify the extension history data.

Synopsis

```
void Set*Value* (*type Value*);
```

Description

This group of methods sets data values into the TWSJExtensionHistory class.

Parameters

`Value`
The data item being set.

Table 3: Mutator Methods for TWSJExtensionHistory Class

Method	Data Parameter	Description
`SetCreateTime`	`const TDate& date`	When a particular extension was created
`SetExtensionDays`	`const TDate& days`	Days in the extension (in seconds)
`SetExtensionHistoryID`	`const GUIDString& id`	GUID for this extension history
`SetExtensionType`	`const TString& type`	EXT (expiration); IFP (Initial Free Period); UNK (unknown)
`SetProductID`	`const GUIDString& id`	GUID for the product belonging to buyer
`SetSubscriberID`	`const GUIDString& id`	GUID for the buyer
`SetSubscriptionID`	`const GUIDString& id`	GUID for the product instance

Return Values

None.

TWSJExtensionHistoryService Class

Name

`TWSJExtensionHistoryService` – The service class for the extension history class.

Synopsis (Constructor and Destructor)

```
TWSJExtensionHistoryService () {};         // constructor
virtual ~TWSJExtensionHistoryService () {};// destructor
```

Description

Methods in this class look up and insert extension history information in the database.

Public Methods

```
DeleteExtensionHistory
InsertExtensionHistory
LookupExtensionsByBuyerID

static Bool Test();
```

Related Classes

`TWSJExtensionHistory`

TWSJExtensionHistoryService::DeleteExtensionHistory

Name

`DeleteExtensionHistory` – Delete extension history objects.

Synopsis

```
HRESULT DeleteExtensionHistory(
                        const VWSJContext* ctxPtr,
                        const GUIDString& id);
```

Description

This method deletes an ExtensionHistory object from the class.

Parameters

`ctxPtr`

Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`id`

Input. A pointer to the TWSJExtensionHistory object to delete from the class.

Return Values

The `DeleteExtensionHistory` method returns:

`S_OK`

If the method was successful.

`E_FAIL`

If the method was not successful.

`<DB Errors>`

If the method resulted in database errors.

TWSJExtensionHistoryService::InsertExtensionHistory

Name

`InsertExtensionHistory` – Insert an extension history object.

Synopsis

```
HRESULT InsertExtensionHistory(
                    const VWSJContext* ctxPtr,
                    TWSJExtensionHistory& suspension);
```

Description

This method inserts an ExtensionHistory object into the class.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`suspension`
: Input. A pointer to a TWSJExtensionHistory object to insert into the class.

Return Values

The `InsertExtensionHistory` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJExtensionHistoryService::LookupExtensionsByBuyerID

Name

`LookupExtensionsByBuyerID` – Look up an extension history object.

Synopsis

```
HRESULT LookupExtensionsByBuyerID(
                const VWSJContext* ctxPtr,
                const GUIDString& buyerID,
                TWSJExtensionVector& extensions);
```

Description

This method looks up an ExtensionHistory object in the database according to the buyer's ID.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyerID`
: Input. A search string to get extension history objects from the database.

`extensions`
: Output. A vector of TWSJExtensionHistory objects for this buyer.

Return Values

The `LookupExtensionsByBuyerID` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJExtensionHistoryService::LookUpByID

Name

`LookUpByID` – Look up an extension history object by GUID.

Synopsis

```
HRESULT LookUpByID(
                const VWSJContext* ctxPtr,
                const GUIDString& id,
                TWSJExtension& extension);
```

Description

This method looks up an ExtensionHistory object in the database according to a GUID.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`id`
: Input. A GUID to get an extension history object from the database.

`extension`
: Output. A TWSJExtensionHistory object.

Return Values

The `LookUpByID` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJFoundationFactory Class

Name

`TWSJFoundationFactory` – The WSJ foundation factory object.

Synopsis (Constructor and Destructor)

```
TWSJFoundationFactory () {};           // constructor
virtual ~TWSJFoundationFactory () {};// destructor
```

Description

Methods in this class create foundation objects. These support the WSJ interfaces to Context objects and Buyer, Subscription, Offer, Product, Group, and Domain objects (and their respective Services).

Public Methods

```
MakeVWSJBuyer
MakeVWSJBuyerService
MakeVWSJContext
MakeVWSJDomain
MakeVWSJDomainService
MakeVWSJGroup
MakeVWSJGroupService
MakeVWSJOffer
MakeVWSJOfferService
MakeVWSJProduct
MakeVWSJProductService
MakeVWSJSubscription
MakeVWSJSubscriptionService
```

Related Classes

TWSJFoundationFactory::MakeVWSJ*Object*

Name

`MakeVWSJ`*`Object`* – Create a foundation object.

Synopses

```
virtual VWSJBuyer* MakeVWSJBuyer ();
virtual VWSJBuyerService* MakeVWSJBuyerService ();
virtual VWSJContext* MakeVWSJContext ();
virtual VWSJDomain* MakeVWSJDomain ();
virtual VWSJDomainService* MakeVWSJDomainService ();
virtual VWSJGroup* MakeVWSJGroup ();
virtual VWSJGroupService* MakeVWSJGroupService ();
virtual VWSJOffer* MakeVWSJOffer ();
virtual VWSJOfferService* MakeVWSJOfferService ();
virtual VWSJProduct* MakeVWSJProduct ();
virtual VWSJProductService* MakeVWSJProductService ();
virtual VWSJSubscription* MakeVWSJSubscription ();
virtual VWSJSubscriptionService* MakeVWSJSubscriptionService ();
```

Description

These methods create VWSJ objects.

Parameters

None

Return Values

The `MakeVWSJ`*`Object`* methods return a pointer to the object created.

Example

```
TWSJFoundationFactory factory;
VWSJBuyer *buyer = factory.MakeVWSJBuyer();
```

TWSJPmtAccountHistory Class

Name

`TWSJPmtAccountHistory` – The WSJ payment history class.

Synopses (Constructors and Destructor)

```
TWSJPmtAccountHistory () {};                      // constructor
TWSJPmtAccountHistory (const TWSJPmtAccountHistory&)){};
                                                    // copy constructor
~TWSJPmtAccountHistory() {};                      // destructor
```

Description

Methods in this class access and change properties related to the history of payments to an account.

Operators

```
Bool operator==(const TWSJPmtAccountHistory&) const;// equality
Bool operator!=(const TWSJPmtAccountHistory&) const;// inequality
TWSJPmtAccountHistory& operator=(const TWSJPmtAccountHistory&);
                                                          // assignment
friend ostream& operator<<(
                 ostream& ost,
                 const TWSJPmtAccountHistory& t);// insertion
```

Public Methods

Get*Data*
Set*Data*

```
static Bool Test();
```

Related Classes

`TWSJPmtAccountHistoryService`

TWSJPmtAccountHistory::Get*Data*

Name

`Get`*`Data`* – Access payment history data.

Synopsis

```
type GetValue () const;
```

Description

This group of methods accesses data values in the class. The values are related to the credit card (or other payment type) used to pay for the account.

Parameters

None

Table 4: Accessor Methods for TWSJPmtAccountHistory Class

Return Type	Method	Description and Notes *
`TString`	`GetName`	Name on the card (the owner's name)
`GUIDString`	`GetPmtAccountID`	GUID of Transact payment account
`GUIDString`	`GetPmtBrandID`	GUID of the payment brand (VI, MC, DC, DI, AM)
`OMKT_EPmtClass`	`GetPmtClass`	Type of payment class: kPmtClassCreditCard, kPmtClassTestCard, kPmtClassSET, kPmtClassDebitCard, kPmtClassSmartCard, kPmtClassRemoteBill, kPmtClassPurchaseOrder
`TString`	`GetPmtInstrumentInfo`	Brand-specific data about the card, usually number, expiration date, type, etc.
`GUIDString`	`GetPrincipalID`	GUID for buyer
`OMKT_EPmtAccount\ Status`	`GetState`	Status: kPmtAccountActive, kPmtAccountDeleted, kPmtAccountInitiated, kPmtAccountSuspended

* See *Payment API* for more information. The enums are from the PmtBase.h file.

TWSJPmtAccountHistory::Set*Data*

Name

Set*Data* – Specify the value of a payment account history data item.

Synopsis

```
void SetValue (type Value);
```

Description

This group of methods sets payment account data into the class.

Parameters

`Value`
The data item being set.

Table 5: Mutator Methods for TWSJPmtAccountHistory Class

Method	Data Parameter	Description
`SetName`	`const TString& name`	Name on the card (the owner's name)
`SetPmtAccountID`	`const GUIDString& id`	GUID of Transact payment account*
`SetPmtBrandID`	`const GUIDString& id`	GUID of the brand (VI, MC, DC, DI, AM)
`SetPmtClass`	`const OMKT_EPmtClass& in`	kPmtClassCreditCard, kPmtClassTestCard, kPmtClassSET, kPmtClassDebitCard, kPmtClassSmartCard, kPmtClassRemoteBill, kPmtClassPurchaseOrder
`SetPmtInstrument\ Info`	`const TString& info`	Expiration date
`SetPrincipalID`	`const GUIDString& id`	GUID for buyer
`SetState`	`const OMKT_EPmtAccountStatus& state`	Status: kPmtAccountActive, kPmtAccountDeleted, kPmtAccountInitiated, kPmtAccountSuspended.

* See *Payment API* for more information. The enums are from the PmtBase.h file.

TWSJProductHistory Class

Name

`TWSJProductHistory` – The WSJ product history class for a buyer.

Synopsis (Constructor and Destructor)

```
TWSJProductHistory () {};                    // constructor
TWSJProductHistory (
              const TWSJProductHistory&)) {};   // copy constructor
virtual ~TWSJProductHistory() {};            // destructor
```

Description

Methods in this class access and change properties such as a subscriber's, periodical, and product ID; and any extensions to the subscription of a particular buyer.

Enumerated Data

```
enum EWSJProdStatus {
   kUnKnown,
   kCurrent,
   kFrozen,
   kCancelled };
```

Table 6: Meanings of Product Status

Value of EWSJProdStatus	Meaning of Status
`kUnKnown`	Unknown
`kCurrent`	Current
`kFrozen`	Frozen
`kCancelled`	Cancelled

Operators

```
Bool operator==(const TWSJProductHistory&) const;// equality
Bool operator!=(const TWSJProductHistory&) const;// inequality
```

```
TWSJProductHistory& operator=(
                        const TWSJProductHistory&); // assignment
friend ostream& operator<<(
                 ostream& ost,
                 const TWSJProductHistory& t);// insertion
```

Public Methods

Get*Data*
Set*Data*

```
Bool IsValid() const;
```

Related Classes

```
TWSJProductHistoryService
```

TWSJProductHistory::Get*Data*

Name

`Get*Data*` – Access the specified product history data for a buyer.

Synopsis

```
virtual type GetValue () const;
```

Description

This group of methods accesses data values in the class related to the subscription (status, price, extension period, name, buyer, etc.) for a specific buyer.

Parameters

None

Table 7: Accessor Methods for TWSJProductHistory Class

Return Type	Method	Description and Notes
`TMoney`	`GetAmount`	Price at which product was offered
`TString`	`GetChargeType`	Subscription or credit
`TDate`	`GetCreateTime`	When record was created
`TDate`	`GetExpireDate`	When subscription expires
`TString`	`GetExtendedFlag`	Was it extended? Is any extension in the history table? Y or N
`TString`	`GetNotes`	Free-form text for this record
`TDate`	`GetOccurredOn`	Date charge occurred
`TString`	`GetPlan`	Name of product that was purchased
`GUIDString`	`GetProductHistoryID`	GUID for this record
`TDate`	`GetStartDate`	When subscription began
`TWSJProduct\ History:: EWSJProdStatus`	`GetStatus`	Status of this subscription: kUnKnown, kCurrent, kFrozen, kCancelled
`GUIDString`	`GetSubscriberID`	GUID for buyer in database

TWSJProductHistory::Set*Data*

Name

Set*Data* – Specify the value of a product history data item .

Synopsis

```
void Set Value (type Value);
```

Description

This group of methods sets product history data values into the class.

Parameters

`Value`
The data item being set.

Table 8: Mutator Methods for TWSJProductHistory Class

Method	Data Parameter	Description
`SetAmount`	`const TMoney& amt`	Price at which product was offered
`SetChargeType`	`const TString& type`	Subscription or credit:
`SetCreateTime`	`const TDate& date`	When record was created
`SetExpireDate`	`const TDate& date`	When subscription expires
`SetExtendedFlag`	`const TString& flag`	Was it extended? Is any extension in the history table? Y or N
`SetNotes`	`const TString& notes`	Free-form text for this record
`SetOccurredOn`	`const TDate& date`	Date charged occurred
`SetPlan`	`const TString& plan`	Name of product that was purchased
`SetProductHistoryID`	`const GUIDString& id`	GUID of this record
`SetStartDate`	`const TDate& date`	When subscription began
`SetStatus`	`const TWSJProductHistory:: EWSJProdStatus status`	Status of this subscription: kUnKnown, kCurrent, kFrozen, kCancelled
`SetSubscriberID`	`const GUIDString& id`	GUID of buyer

TWSJProductHistoryService Class

Name

`TWSJProductHistoryService` – Insert product history objects for a buyer.

Synopsis (Constructor and Destructor)

```
TWSJProductHistoryService () {};          // constructor
virtual ~TWSJProductHistoryService () {};// destructor
```

Description

Methods in this class look up and insert product history information for a specific buyer in the database.

Public Methods

```
InsertProductHistory
LookupProductHistoryByBuyerID

static Bool Test();
```

Related Classes

`TWSJProductHistory`

TWSJProductHistoryService::InsertProductHistory

Name

`InsertProductHistory` – Insert a product history object into the database for a buyer.

Synopsis

```
HRESULT InsertProductHistory(
                    const VWSJContext* ctxPtr,
                    TWSJProductHistory& productHistory);
```

Description

This method inserts a ProductHistory object into the database for a Buyer.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`productHistory`
: Input. The TWSJProductHistory object to insert into the database.

Return Values

The `InsertProductHistory` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJProductHistoryService::LookupProductHistoryByBuyerID

Name

`LookupProductHistoryByBuyerID` – Look up product history object.

Synopsis

```
HRESULT LookupProductHistoryByBuyerID(
                const VWSJContext* ctxPtr,
                const GUIDString& buyerID,
                TWSJProductHistoryVector& productHistories);
```

Description

This method looks up product history objects according to the buyer's ID.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyerID`
: Input. A search string to get suspension IDs from the database.

`productHistories`
: Output. A vector of TWSJProductHistory objects for this buyer.

Return Values

The `LookupProductHistoryByBuyerID` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJProductOffer Class

Name

`TWSJProductOffer` – The WSJ product offer class.

Synopsis (Constructor and Destructor)

```
TWSJProductOffer () {};                      // constructor
TWSJProductOffer (
              const TWSJProductOffer&)) {};   // copy constructor
~TWSJProductOffer() {};                      // destructor
```

Description

Methods in this class access and change IDs for products and offers. This is how a product is associated with an offer.

Operators

```
Bool operator==(const TWSJProductOffer&) const;// equality
Bool operator!=(const TWSJProductOffer&) const;// inequality
TWSJProductOffer& operator=(
                        const TWSJProductOffer&); // assignment
friend ostream& operator<<(
                 ostream& ost,
                 const TWSJProductOffer& t);// insertion
```

Public Methods

```
Get Data
Set Data

Bool IsValid () const;
```

Related Classes

`TWSJProductOfferService`

TWSJProductOffer::Get*Data*

Name

`Get`*`Data`* – Access the value of the specified product offer data.

Synopsis

```
virtual type GetValue () const;
```

Description

This group of methods accesses data values in the class that map an offer to a set of products.

Parameters

None

Table 9: Accessor Methods for TWSJProductOffer Class

Return Type	Method	Description and Notes
`GUIDString`	`GetOfferID`	Offer GUID
`GUIDString`	`GetProductID`	Product GUID
`GUIDString`	`GetProductOfferID`	Product offer GUID

Example

Notes

There may be one offer, but many products.

TWSJProductOffer::Set*Data*

Name

`Set`*`Data`* – Specify the value of the product offer data item.

Synopsis

```
void Set Value (type Value);
```

Description

This group of methods sets data values into the class to map an offer to a set of products.

Parameters

`Value`
: The data item being set.

Table 10: Mutator Methods for TWSJProductOffer Class

Method	Data Parameter	Description
`SetOfferID`	`const GUIDString& id`	GUID of offer
`SetProductID`	`const GUIDString& id`	GUID of product for this offer
`SetProductOfferID`	`const GUIDString& id`	GUID of this product offer

Return Values

None.

Notes

There may be one offer, but many products.

TWSJProductOfferService Class

Name

`TWSJProductOfferService` – The WSJ ProductOfferService class.

Synopsis (Constructor and Destructor)

```
TWSJProductOfferService () {};            // constructor
~TWSJProductOfferService () {};           // destructor
```

Description

Methods in this class look up and insert product offer information in the database.

Public Methods

```
InsertProductOffer
LookUpByID
UpdateProductOffer

static Bool Test();
```

Related Classes

`TWSJProductOffer`

TWSJProductOfferService::InsertProductOffer

Name

`InsertProductOffer` – Insert a product offer object into the database.

Synopsis

```
HRESULT InsertProductOffer(
                    const TSubsysContext& context,
                    TWSJProductOffer& productOffer);
```

Description

This method inserts a ProductOffer object into the database.

Parameters

`context`
: Input. A pointer to a TSubsysContext object to specify the object containing the operational context required to perform database operations.

`productOffer`
: Input. The TWSJProductOffer object to insert into the database.

Return Values

The `InsertProductOffer` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJProductOfferService::LookUpByID

Name

`LookUpByID` – Look up a product offer object.

Synopsis

```
HRESULT LookUpByID(
                const TSubsysContext* context,
                const GUIDString& buyerID,
                TWSJProductOffer& productOffer);
```

Description

This method looks up ProductOfferService objects in the database according to the buyer's ID.

Parameters

`context`
: Input. A pointer to a TSubsysContext object to specify the object containing the operational context required to perform database operations.

`buyerID`
: Input. A search string to get product offers from the database.

`productOffer`
: Output. The TWSJProductOffer object for this buyer.

Return Values

The `LookUpByID` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJProductOfferService::UpdateProductOffer

Name

`UpdateProductOffer` – Update a product offer object.

Synopsis

```
HRESULT UpdateProductOffer(
                    const TSubsysContext& context,
                    TWSJProductOffer& productOffer);
```

Description

This method inserts a ProductOffer object into the database.

Parameters

`context`
: Input. A pointer to a TSubsysContext object to specify the object containing the operational context required to perform database operations.

`productOffer`
: Input. The TWSJProductOffer object to update in the database.

Return Values

The `UpdateProductOffer` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJSearchCriteria Class

Name

`TWSJSearchCriteria` – The WSJ Search criteria class.

Synopsis (Constructor and Destructor)

```
TWSJSearchCriteria () {};                    // constructor
TWSJSearchCriteria (
              const TWSJSearchCriteria&)) {};   // copy constructor
virtual ~TWSJSearchCriteria() {};            // destructor
```

Description

Methods in this class access and change properties related to a subscriber's account data, such as address, activity, cost, email, names, etc.

Operators

```
Bool operator==(const TWSJSearchCriteria& rhs) const;// equality
Bool operator!=(const TWSJSearchCriteria& rhs) const;// inequality
TWSJSearchCriteria& operator=(
                        const TWSJSearchCriteria&); // assignment
```

Public Methods

Get*Data*
Set*Data*

Related Classes

`TWSJ. . .`

TWSJSearchCriteria::Get*Data*

Name

`Get`*`Data`* – Access the value of the specified search criterion.

Synopsis

```
virtual type GetValue () const;
```

Description

This group of methods accesses data values in the class in order to find a subscriber. Much of this information was supplied by the subscriber upon registration.

Parameters

None

Table 11: Accessor Methods for TWSJSearchCriteria Class

Return Type	Method	Description and Notes
`Bool`	`GetActive`	Buyer's status is active?
`TString`	`GetAreaCode`	Buyer's telephone area code
`Bool`	`GetBillable`	Buyer uses credit card or purchase plan?
`Bool`	`GetCancelled`	Buyer is canceled?
`TString`	`GetCompany`	Name of buyer's company
`Bool`	`GetComplimentary`	Subscription is complimentary?
`TString`	`GetDJAccount`	DJ account number (not known by buyer)
`TString`	`GetEmail`	Buyer's email address
`TString`	`GetFirstName`	Buyer's first name
`Bool`	`GetFree`	Buyer's account is free, i.e., corporate?
`Bool`	`GetFrozen`	Buyer cannot access site's resources?
`TString`	`GetGroup`	Name of group to which buyer subscribes
`Bool`	`GetInIFP`	Buyer is in Initial Free Period?

Table 11: Accessor Methods for TWSJSearchCriteria Class

Return Type	Method	Description and Notes
`TString`	`GetLastName`	Buyer's last name
`TString`	`GetLoginName`	Buyer's access name when registering
`TString`	`GetMaxHits`	Maximum number of records to return on the query for this search
`Bool`	`GetResourceSuspension`	Buyer is in resource-suspended state?
`TString`	`GetState`	Buyer's state of residence, e.g., Utah

Example

TWSJSearchCriteria::Set*Data*

Name

`Set`*`Data`* – Specify the value of the search criteria data item.

Synopsis

```
void Set Value (type Value);
```

Description

This group of methods sets data values into the class related to search criteria in order to change such data as a buyer's contact information, status, etc.

Parameters

`Value`
: The data item being set.

Table 12: Mutator Methods for TWSJSearchCriteria Class

Method	Data Parameter	Description
`SetActive`	`const Bool& in`	Set buyer's status to active?
`SetAreaCode`	`const TString& areacode`	Buyer's telephone area code
`SetBillable`	`const Bool& in`	Buyer uses credit card or purchase plan?
`SetCancelled`	`const Bool& in`	Buyer is canceled?
`SetCompany`	`const TString& company`	Name of buyer's company
`SetComplimentary`	`const Bool& in`	Subscription is complimentary?
`SetDJAccount`	`const TString& account`	DJ account number (not known by buyer)
`SetEmail`	`const TString& email`	Buyer's email address
`SetFirstName`	`const TString& first`	Buyer's first name
`SetFree`	`const Bool& in`	Buyer's account is corporate or free?
`SetFrozen`	`const Bool& in`	Buyer cannot access site's resources?
`SetGroup`	`const TString& group`	Buyer's group affiliation(s)
`SetInIFP`	`const Bool& in`	Buyer is in Initial Free Period?
`SetLastName`	`const TString& last`	Buyer's last name
`SetLoginName`	`const TString& name`	Buyer's access name when registering
`SetMaxHits`	`const TString& maxhits`	Maximum number of database hits (buyers) to show for this search
`SetResource\ Suspension`	`const Bool& in`	Buyer is in resource-suspended state?
`SetState`	`const TString& state`	Buyer's state of residence

Return Values

None.

TWSJSuspensionHistory Class

Name

`TWSJSuspensionHistory` – The WSJ suspension history class.

Synopsis (Constructor and Destructor)

```
TWSJSuspensionHistory () {};                // constructor
TWSJSuspensionHistory (
              const TWSJSuspensionHistory&)) {};   // copy constructor
virtual ~TWSJSuspensionHistory() {};        // destructor
```

Description

Methods in this class access and change properties related to a buyer's suspension history: subscriber and periodical ID; status, suspension date and history, comments, etc.

Operators

```
Bool operator==(const TWSJSuspensionHistory&) const;// equality
Bool operator!=(const TWSJSuspensionHistory&) const;// inequality
TWSJSuspensionHistory& operator=(
                        const TWSJSuspensionHistory&); // assignment
friend ostream& operator<<(
                  ostream& ost,
                  const TWSJSuspensionHistory& t);// insertion
```

Public Methods

```
Get Data
Set Data

Bool IsValid() const;
```

Related Classes

`TWSJSuspensionHistoryService`

TWSJSuspensionHistory::Get*Data*

Name

`Get*Data*` – Access the value of the specified suspension history data.

Synopsis

```
virtual type GetValue () const;
```

Description

This group of methods accesses data values in the class related to the subscription and its suspension characteristics.

Parameters

None

Table 13: Accessor Methods for TWSJSuspensionHistory Class

Return Type	Method	Description and Notes
`TString`	`GetComment`	Free-form text
`TString`	`GetCSR`	Initials of Customer Service Representative
`GUIDString`	`GetPeriodicalID`	GUID of periodical (resource)
`GUIDString`	`GetSubscriberID`	GUID of buyer
`VWSJSubscription ::EWSJSubType`	`GetSubType`	Subscription type: kNoSub, kSingle, kComp, kFree
`TDate`	`GetSuspensionDate`	When resource was suspended
`GUIDString`	`GetSuspensionHistoryID`	GUID of this record
`TString`	`GetSuspensionStatus`	A (activated) or S (suspended)

TWSJSuspensionHistory::Set*Data*

Name

`Set`*`Data`* – Specify the value of the suspension history data item.

Synopsis

```
void Set Value (type Value);
```

Description

This group of methods sets data values into the class to specify suspension-related data: date suspended; change of status; periodical being suspended or activated; the buyer's Customer Service Representative.

Parameters

`Value`
: The data item being set.

Table 14: Mutator Methods for TWSJSuspensionHistory Class

Method	Data Parameter	Description
`SetComment`	`const TString& comment`	Free-form text
`SetCSR`	`const TString& csr`	Initials of Customer Service Representative
`SetPeriodicalID`	`const GUIDString& id`	GUID of periodical (resource)
`SetSubscriberID`	`const GUIDString& id`	GUID of buyer
`SetSubType`	`const VWSJSubscription:: EWSJSubType& subtype`	Subscription type: kNoSub, kSingle, kComp, kFree
`SetSuspensionDate`	`const TDate& date`	When resource was suspended
`SetSuspensionHistoryID`	`const GUIDString& id`	GUID of this record
`SetSuspensionStatus`	`const TString& status`	A (activated) or S (suspended)

Return Values

None.

TWSJSuspensionHistoryService Class

Name

`TWSJSuspensionHistoryService` – The WSJ suspension history object.

Synopsis (Constructor and Destructor)

```
TWSJSuspensionHistoryService () {};        // constructor
virtual ~TWSJSuspensionHistoryService () {};// destructor
```

Description

Methods in this class look up and insert suspension history information in the database.

Public Methods

```
InsertSuspensionHistory
LookupSuspensionsByBuyerID

static Bool Test();
```

Related Classes

`TWSJSuspensionHistory`

TWSJSuspensionHistoryService::InsertSuspensionHistory

Name

`InsertSuspensionHistory` – Insert a suspension history object.

Synopsis

```
HRESULT InsertSuspensionHistory(
                    const VWSJContext* ctxPtr,
                    TWSJSuspensionHistory& suspension);
```

Description

This method inserts a SuspensionHistory object into the database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`suspension`
: Input. The TWSJSuspensionHistory object to insert into the database.

Return Values

The `InsertSuspensionHistory` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

TWSJSuspensionHistoryService::LookupSuspensionsByBuyerID

Name

`LookupSuspensionsByBuyerID` – Look up a suspension history object.

Synopsis

```
HRESULT LookupSuspensionsByBuyerID(
                const VWSJContext* ctxPtr,
                const GUIDString& buyerID,
                TWSJSuspensionVector& suspensions);
```

Description

This method looks up SuspensionHistory objects in the database according to the buyer's ID.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyerID`
: Input. A search string to get suspension IDs from the database.

`suspensions`
: Output. A vector of TWSJSuspensionHistory objects for this buyer.

Return Values

The `LookupSuspensionHistoryByBuyerID` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJBuyer Class

Name

`VWSJBuyer` – The WSJ buyer class.

Description

Methods in this class access and change buyer-related properties such as a buyer's ID, account number, password, name, and subscription-related data.

Enumerated Data

```
enum EWSJRefundType {
   kRefundFull,
   kRefundPartial,
   kRefundChargeback,
   kRefundNone };
```

Table 15: Meanings of Refund Types

Value of EWSJRefundType	Type of Refund
`kRefundFull`	Full
`kRefundPartial`	Partial
`kRefundChargeback`	Charged back to credit card
`kRefundNone`	None

```
enum EWSJRenew {
   kRenew,
   kDoNotRenew };
```

Table 16: Meanings of Renewal Enums

Value of EWSJRenew	Type of Renewal
`kRenew`	Renew
`kDoNotRenew`	Do not renew

Public Methods

```
AddAnswer, AddQuestion
AddToGroup
Get<i>Data</i>
Set<i>Data</i>

virtual Bool IsValid() const=0;
virtual Bool IsChanged() const=0;
```

Related Classes

```
VWSJBuyerService
TWSJBuyerStateHandler
```

VWSJBuyer::AddAnswer, AddQuestion

Name

`AddAnswer` – Add buyer's secret.

`AddQuestion` – Add the question that elicits buyer's secret.

Synopses

```
virtual void AddAnswer(
                          const TString& answer,
                           Bool DirtyFlag = TRUE) =0;
virtual void AddQuestion(
                          const TString& question,
                           Bool DirtyFlag = TRUE) =0;
```

Description

This method adds a buyer's secret and appropriate question.

Parameters

`answer`
: Input parameter. The answer (secret).

`question`
: Input parameter. The question to elicit the secret.

`DirtyFlag`
: Input parameter. If set TRUE, causes the corresponding dirty flag for each dirty member to be set to TRUE (i.e., it is dirty) so the Foundation API can make changes accordingly .

Return Values

None.

VWSJBuyer::AddToGroup

Name

`AddToGroup` – Add the current buyer to a group.

Synopsis

```
virtual HRESULT AddToGroup(
                           VWMSJGroup* group,
                            Bool DirtyFlag = TRUE) =0;
```

Description

This method adds the current buyer to a group.

Parameters

`group`
: Input parameter. The group, specified as a pointer of type `VWSJGroup`, to which to add the buyer.

`DirtyFlag`
: Input parameter. If set TRUE, causes the corresponding dirty flag for each dirty member to be set to TRUE (i.e., it is dirty) so the Foundation API can make changes accordingly .

Return Values

The `AddToGroup` method returns:

`S_OK`
: If the method successfully added the buyer.

`E_FAIL`
: If the method could not successfully add the buyer.

VWSJBuyer::Get*Data*

Name

`Get*Data*` – Access the value of the buyer-related data item.

Synopsis

```
virtual type GetValue () const =0;
```

Description

This group of methods accesses data values in the WSJ database related to information the buyer specifies during registration.

Parameters

None

Table 17: Accessor Methods for VWSJBuyer Class

Return Type	Method	Description and Notes
`TString`	`GetAccessName`	Buyer's chosen login name
`TAddress`	`GetAddress`	Buyer's street address
`HRESULT`	`GetAnnualStockTxs (TString& out)`	Stated number of annual stock transactions
`TOMKTStringVector`	`GetAnswers`	Mother's maiden name or other secret(s)
`HRESULT`	`GetAuthorizedAgent (TString& out)`	Person who can ask questions about or change the account on behalf of the buyer
`HRESULT`	`GetBusiness (TString& out)`	Stated profession or type of business
`GUIDString`	`GetBuyerID`	Buyer's GUID
`HRESULT`	`GetByPassAuth (TString& out)`	By-pass credit card authorization for this buyer
`HRESULT`	`GetBuyerLoginMessage (TString& out)`	What buyer sees on login (set by CSR)
`HRESULT`	`GetCompany (TString& out)`	Name of buyer's company

Table 17: Accessor Methods for VWSJBuyer Class

Return Type	Method	Description and Notes
TDate	GetCreateTime	When this buyer record was created
HRESULT	GetCSRNotes (TString& out)	Custom notes on this buyer as added by CSR
TExtensionData	GetCustomFields	See *Transact Utility Guide*
HRESULT	GetDJ7AccountNumber (TString& out)	Dow Jones internal ID for this buyer in OLF -- may not be set
HRESULT	GetDJPublication (TString& out)	Dow Jones publication the buyer reads, , e.g., *Barron's, Wall Street Journal*
HRESULT	GetDontUseCookies (TString& out)	If true, do not authenticate using cookies; else use cookies and auto-login
TString	GetEmail	Buyer's email address
TString	GetFirstName	Buyer's first name
HRESULT	GetGender (TString& out)	Gender of buyer
HRESULT	GetLastCCName (Tstring& out)	Name on active (last) credit card account
TString	GetLastName	Buyer's last name
HRESULT	GetLastPmtAccount (GUIDString& out)	Get GUID for buyer's active (last) payment account
HRESULT	GetLastSubscription (GUIDString& out)	GUID of active subscriptions
HRESULT	GetMailAlerts (Tstring& out)	List of email alerts that buyer chose: alerts and new features
HRESULT	GetMarkForPurgedDate (Tstring& out)	Date when buyer was first marked for purge
HRESULT	GetNewsAlerts (Tstring& out)	List of which news alerts the buyer chose: business, market, and/or tech news
HRESULT	GetOfferID (GUIDString& out)	GUID of offer being bought
HRESULT	GetOLFAccountNumber (GUIDString& out)	OLF account number for this buyer (set by OLF system)
HRESULT	GetOrganizationSize (TString& out)	Buyer's stated size of organization (a.k.a. profession, business, company)
TString	GetPassword	Buyer's chosen password
GUIDString	GetPmtAccount	Get GUID for buyer's active (last) payment account

Table 17: Accessor Methods for VWSJBuyer Class

Return Type	Method	Description and Notes
`HRESULT`	`GetPrintAccountNumber (TString& out)`	Acct number for hardcopy subscription (if one exists for this buyer)
`HRESULT`	`GetProfession (TString& out)`	Buyer's selected profession
`TOMKTStringVector`	`GetQuestions`	Question(s) for buyer's secret(s)
`HRESULT`	`GetReaderFrequency (TString& out)`	Selected frequency to say how often buyer reads *Wall Street Journal*
`TOMKTStringVector`	`GetRequestedDomains`	Resources requested for buyer access; to be restricted, activated, etc.
`TDate`	`GetRequestedExtendedDays`	Number of days requested to extend IFP of subscription
`TDate`	`GetRequestedExtended\ ExpireDays`	Number of days requested to extend subscription
`GUIDString`	`GetRequestedExtension\ Delete`	Delete extension history record associated with this GUID
`TWSJGroupVector`	`GetRequestedGroups`	Groups that buyer is requesting to join or freeze
`VWSJPmtAccount*`	`GetRequestedPmtAccount`	Requested active payment account for this buyer
`EWSJRefundType`	`GetRequestedRefund`	Get refund (full or none)
`VWSJBuyer::\ EWSJRenew`	`GetRequestedRenew`	Return kRenew, kDoNotRenew
`VWSJSubscription*`	`GetRequestedSubscription`	Requested active payment account for this buyer
`HRESULT`	`GetSubscriberStatus (TString& out)`	Status can be: active, activeIFP, cancelled, cancelledIFP, frozen, noAccess, purged, unknown.
`HRESULT`	`GetSubscriptionType (int& out)`	Subscription type: kNoSub,kSingle, kComp, kFree
`HRESULT`	`GetUseSSL (TString& out)`	Not currently used
`HRESULT`	`GetYearBorn (TString& out)`	Year buyer was born, as stated at registration

Example

VWSJBuyer::Set*Data*

Name

`Set*Data*` – Specify the value of the buyer-related data item.

Synopses

```
virtual { void | HRESULT } Set Value (type Value,
       Bool DirtyFlag = TRUE
       ) =0;

virtual ReturnType SetValue ( ) const = 0;
```

Description

This group of methods sets data values into the WSJ database primarily for registration-related data.

Parameters

`Value`
: The data item being set.

`DirtyFlag`
: If set TRUE, causes the corresponding dirty flag for each dirty member to be set to TRUE (i.e., it is dirty) so the Foundation API can make changes accordingly .

Table 18: Mutator Methods for VWSJBuyer Class

Method (and Return Type If not Void)	Data Parameter	Description
`SetAccessName`	`const TString& accessname`	Buyer's chosen login name
`SetAddress`	`const TAddress& address`	Buyer's address
`SetAnnualStockTxs (HRESULT)`	`const TString& in`	Stated number of annual stock transactions

Table 18: Mutator Methods for VWSJBuyer Class

Method (and Return Type If not Void)	Data Parameter	Description
`SetAuthorizedAgent (HRESULT)`	`const TString& in`	Agent authorized to retrieve and change buyer's account information
`SetBusiness (HRESULT)`	`const TString& in`	Stated profession or business
`SetBuyerID`	`const GUIDString& id`	GUID for this buyer
`SetBuyerLoginMessage (HRESULT)`	`const TString& in`	To display on login
`SetByPassAuth (HRESULT)`	`const TString& in`	Whether to do a $1 authorization against the payment agent
`SetCompany (HRESULT)`	`const TString& in`	Name of buyer's company
`SetCSRNotes (HRESULT)`	`const TString& in`	Custom notes from CS Rep
`SetCustomFields`	`const TExtensionData& extData`	See *Transact 4 Utility Classes API*
`SetDJPublication (HRESULT)`	`const TString& in`	Buyer's hardcopy subscriptions
`SetDontUseCookies (HRESULT)`	`const TString& in`	If true, do not authenticate using cookies; else use cookies and auto-login
`SetEmail`	`const TString& email`	Buyer's email address
`SetFirstName`	`const TString& firstname`	Buyer's first name
`SetGender (HRESULT)`	`const TString& in`	Buyer's gender
`SetLastCCName (HRESULT)`	`const TString& in`	Name on active (last) credit card
`SetLastName`	`const TString& lastname`	Buyer's last name
`SetMailAlerts (HRESULT)`	`const TString& in`	Mail alerts chosen by buyer at registration
`SetNewsAlerts (HRESULT)`	`const TString& in`	News alerts chosen by buyer at registration
`SetOfferID (HRESULT)`	`const TString& in`	GUID of this offer
`SetOLFAccountNumber (HRESULT)`	`const TString& in`	Set OLF account number (generated by OLF system)
`SetOrganizationSize (HRESULT)`	`const TString& in`	Stated size of buyer's organization, business, or profession
`SetPassword`	`const TString& password`	Buyer's chosen password
`SetPrintAccountNumber (HRESULT)`	`const TString& number`	Acct number from hardcopy subscription (if one exists)

Table 18: Mutator Methods for VWSJBuyer Class

Method (and Return Type If not Void)	Data Parameter	Description
`SetProfession (HRESULT)`	`const TString& in`	Buyer's selected profession or business
`SetReaderFrequency (HRESULT)`	`const TString& in`	Buyer's selected rate of reading *Wall Street Journal*
`SetRequestedDomains`	`const TOMKTStringVector& domains`	Resources requested for buyer access; to be restricted, activated, etc.
`SetRequestedExtendedDays`	`const TDate& days`	No. days requested to extend subscription
`SetRequestedExtended\ ExpireDays`	`const TDate& days`	Number of days requested to extend subscription
`SetRequestedExtension\ Delete`	`const GUIDString& extID`	Delete extension history record associated with this GUID
`SetRequestedRefund`	`const VWSJBuyer:: EWSJRefundType refundType`	Requested type of refund; see Table 15 on page 73
`SetRequestedPmtAccount`	`const VWSJPmtAccount* pmtAcct`	Requested active (last) payment account for this buyer
`SetRequestedRenew`	`const VWSJBuyer:: EWSJRenew refundFlag`	Requested type of renewal; see Table 16 on page 73
`SetRequestedSubscription`	`const VWSJSubscription* sub`	Requested active payment account for this buyer
`SetSubscriberStatus`	`const TString& in`	Set buyer status: kPmtAccountActive, kPmtAccountDeleted, kPmtAccountInitiated, kPmtAccountSuspended (from Pmtbase.h)
`SetSubscriptionType`	`const VWSJSubscription:: EWSJSubType in,`	Set subscription type: kNoSub,kSingle, kComp, kFree
`SetUseSSL (HRESULT)`	`const TString& in`	Not currently used
`SetYearBorn (HRESULT)`	`const TString& in`	Stated year buyer was born

Notes

In general, a SetRequested method needs to call the TWSJBuyerStateHandler class to make a commit to the database.

VWSJBuyerExtra Class

Name

`VWSJBuyerExtra` – Custom, state-relevant field data for a specific buyer.

Description

`VWSJBuyerExtra` is a paper object that reflects a row in the wsjie_buyer_extra table in the database. This paper object is read-only and cannot modify the contents of the table.

Methods in this class access buyer-related properties such as a buyer's ID, account number, password, name, and subscription-related data. They provide a performance improvement over accessing these properties via VWSJBuyer class methods.

Public Methods

`Get`*`Data`*

```
virtual Bool IsValid() const=0;
virtual Bool IsChanged() const=0;
```

Related Classes

`VWSJBuyer`

`VWSJBuyerService`

`VWSJBuyerExtraService`

VWSJBuyerExtra::Get*Data*

Name

`Get`*`Data`* – Access the value of the buyer-related data item.

Synopsis

```
virtual type GetValue () const =0;
```

Description

This group of methods accesses data values in the WSJ database related to information the buyer specifies during registration.

Parameters

None

Table 19: Accessor Methods for VWSJBuyerExtra Class

Return Type	Method	Description and Notes
`TString`	`GetAnnualStockTxs`	Stated number of annual stock transactions
`TString`	`GetAuthorizedAgent`	Person who can ask questions about or change the account on behalf of the buyer
`TString`	`GetBusiness`	Stated profession or type of business
`GUIDString`	`GetBuyerID`	Buyer's GUID
`TString`	`GetBuyerLoginMessage`	What buyer sees on login (set by CSR)
`TString`	`GetByPassAuth`	By-pass credit card authorization for this buyer
`TString`	`GetCompany`	Name of buyer's company
`TString`	`GetCSRNotes`	Custom notes on this buyer as added by CSR
`TString`	`GetDJ7AccountNumber`	Dow Jones internal ID for this buyer in OLF -- may not be set
`TString`	`GetDJPublication`	Dow Jones publication the buyer reads, , e.g., *Barron's, Wall Street Journal*

Table 19: Accessor Methods for VWSJBuyerExtra Class

Return Type	Method	Description and Notes
`TString`	`GetDontUseCookies`	If true, do not authenticate using cookies; else use cookies and auto-login
`TString`	`GetGender`	Gender of buyer
`TString`	`GetLastCCName`	Name on active (last) credit card account
`TString`	`GetLastName`	Buyer's last name
`TString`	`GetLastPmtAccount`	Get GUID for buyer's active (last) payment account
`TString`	`GetLastSubscription`	GUID of active subscriptions
`TString`	`GetMailAlerts`	List of email alerts that buyer chose: alerts and new features
`TString`	`GetMarkForPurgedDate`	Date when buyer was first marked for purge
`TString`	`GetNewsAlerts`	List of which news alerts the buyer chose: business, market, and/or tech news
`TString`	`GetOfferID`	GUID of offer being bought
`TString`	`GetOLFAccountNumber`	OLF account number for this buyer (set by OLF system)
`TString`	`GetOrganizationSize`	Buyer's stated size of organization (a.k.a. profession, business, company)
`HRESULT`	`GetPassword (TString& out)`	Buyer's password
`TString`	`GetPrintAccountNumber`	Acct number for hardcopy subscription (if one exists for this buyer)
`TString`	`GetProfession`	Buyer's selected profession
`TString`	`GetReaderFrequency`	Selected frequency to say how often buyer reads *Wall Street Journal*
`TString`	`GetSubscriberStatus`	Status can be: active, activeIFP, cancelled, cancelledIFP, frozen, noAccess, purged, unknown.
`VWSJSubscription ::EWSJSubType`	`GetSubscriptionType`	kNoSub,kSingle, kComp, kFree
`TString`	`GetUseSSL`	Not currently used
`TString`	`GetYearBorn`	Year buyer was born, as stated at registration

VWSJBuyerExtraService Class

Name

`VWSJBuyerExtraService` – Operate on VWSJBuyerExtra objects.

Description

The method in this class returns a VWSJBuyerExtra object, providing access to information from the database about or related to the buyer, but with increased performance compared with similar VWSJBuyer methods.

Public Methods

`LookUpBYID`

Related Classes

`VWSJBuyerExtra`

VWSJBuyerExtraService::LookUpByID

Name

`LookUpByID` – Look up VWSJBuyerExtra object.

Synopsis

```
virtual HRESULT LookUpByID(const VWSJContext* vCtxPtr,
                           const GUIDString& id,
                           VWSJBuyerExtra *buyerextra) = 0;
```

Description

The method returns a VWSJBuyerExtra object, providing access to information from the database about or related to the buyer, but with increased performance compared with similar VWSJBuyer methods.

Parameters

`vctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This specifies the object containing the operational context required to perform database operations.

`id`
: Input. A GUIDString.

`buyerextra`
: Output. A pointer to a VWSJBuyerExtra object with the requested ID.

Status Returns

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

VWSJBuyerService Class

Name

`VWSJBuyerService` – Operate on buyer objects.

Description

Methods in this class provide a great deal of information from the database about or related to the buyer: account numbering, status (including suspension, reinstatement, and destruction), payment and subscription information, access history, mail alerts, and on and on.

Public Methods

`Get`*`Data`*
`LookUp`*`Data`*
`Set`*`Data`*
`SuspendBuyerResource`
`UpdateCustomFields`
`UpdatePasswordInTx`
`VerifyChallenge`

Related Classes

`VWSJBuyer`

VWSJBuyerService::Get*Data*

Name

`Get*Data*` – Access the value of the specified data.

Synopsis

```
virtual HRESULT Get BuyerInfo ( . . . ) =0;
```

Description

This group of methods accesses data values in the WSJ database. These values are lists of access history and product history, extension and payment accounts.

Parameters

Several, and varied. See Table 20, "Get Accessor Methods for VWSJBuyerService Class," for their usage.

`ctxPtr`

Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used by all Get methods for the VWSJBuyerService class. It specifies the object containing the operational context required to perform database operations.

`buyer`

Input. A pointer to a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`. This is input to all Get methods for the VWSJBuyerService class.

`<other>`

Most of the VWSJBuyerService methods use a third parameter (and some a fourth) that is method-specific.

Table 20: Get Accessor Methods for VWSJBuyerService Class

Method	Parameters	Description and Notes
`GetAccessHistory\ List`	`const VWSJContext* ctxPtr, const VWSJBuyer* buyer, TWSJAccessVector& accesses`	Return a list of access history objects (currently not employed in system)

Table 20: Get Accessor Methods for VWSJBuyerService Class

Method	Parameters	Description and Notes
`GetExtensionList`	`const VWSJContext* ctxPtr, const VWSJBuyer* buyer, TWSJExtensionVector& extensions`	Return a list of extension history objects
`GetPmtAccountList`	`const VWSJContext* ctxPtr, const VWSJBuyer* buyer, TWSJPmtAccountVector& PmtAccounts`	Return a list of payment account history objects
`GetProductHistory List`	`const VWSJContext* ctxPtr, const VWSJBuyer* buyer, TWSJProductHistoryVector& products`	Return a list of product history objects
`GetSuspensionList`	`const VWSJContext* ctxPtr, const VWSJBuyer* buyer, TWSJSuspensionVector& suspensions`	Return a list of suspension history objects

Status Returns

All the Get methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

VWSJBuyerService::LookUp*Data*

Name

`LookUp`*`Data`* – Look up buyers according to specified data.

Synopsis

```
virtual HRESULT LookUpBuyerInfo ( . . . ) =0;
```

Description

This group of methods looks up buyers according to account numbers, name, ID.

Parameters

Several, and varied. See Table 21, "LookUp Accessor Methods for VWSJBuyerService Class," for their usage.

`ctxPtr`

Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by nearly all LookUp methods for the VWSJBuyerService class. It specifies the object containing the operational context required to perform database operations.

`buyer`

Input and output. A pointer to a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`. This is input to or returned by nearly all LookUp methods for the VWSJBuyerService class.

`<other>`

Most of the LookUp methods use a method-specific third parameter (and some a fourth).

Table 21: LookUp Accessor Methods for VWSJBuyerService Class

Method	Parameters	Description and Notes
`LookUp`	`const VWSJContext* ctxPtr, const TWSJSearchCriteria& query, TGUIDStringVector& buyers`	Look up buyers by search criteria

Table 21: LookUp Accessor Methods for VWSJBuyerService Class

Method	Parameters	Description and Notes
`LookUpBuyer\ byDJAcctNum`	`const VWSJContext* ctxPtr, const TString& djAcctNumber, VWSJBuyer* buyer`	Return buyer object based on DJ account number
`LookUpBuyer\ byName`	`const VWSJContext* ctxPtr, const TString& name, VWSJBuyer* buyer`	Return buyer object based on login name
`LookUpBuyerby\ DJPrintAcctNum`	`const VWSJContext* ctxPtr, const TString& djPrintAcctNumber, VWSJBuyer* buyer`	Return buyer object based on DJ print account number
`LookUpBuyerbyID`	`const VWSJContext* ctxPtr, const GUIDString& id, VWSJBuyer* buyer, Bool DoFullLookUp = TRUE`	Return buyer object based on this buyer GUID. The flag when `TRUE (the default)` returns all the buyer's info.

Status Returns

All the LookUp methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

VWSJBuyerService::Set*Data*

Name

`Set*Data*` – Set mail alert information for a buyer.

Synopsis

```
virtual HRESULT Set*BuyerInfo* ( . . . ) =0;
```

Description

This group of currently one method sets data values into the WSJ database for the types of mail alerts a buyer wants to receive..

Parameters

`ctxPtr`

Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Set methods for the VWSJBuyerService class. It specifies the object containing the operational context required to perform database operations.

`buyer`

Input and output. A pointer to a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`. This is input to or returned by all Set methods for VWSJBuyerService class.

`webdata`

Input. Specifies the criteria to set for mail alerts.

Table 22: Mutator Methods for VWSJBuyerService Class

Method	Parameters	Description and Notes
`SetMailAlerts`	`const VWSJContext* ctxPtr, VWSJBuyer* buyer, const TWebData& webdata`	Set the mail alerts custom field. The `webdata` input specifies the criteria to set.

Status Returns

All the Set methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

VWSJBuyerService::SuspendBuyerResource

`SuspendBuyerResource` – Suspend a buyer from a domain.

Synopsis

```
virtual HRESULT SuspendBuyerResource(  const VWSJContext* ctxPtr,
                            VWSJBuyer* buyer,
                            const TString& domain) = 0;
```

Description

This method suspends a buyer from the specified domain. If the specified domain is null, the buyer will be denied from all domains.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyer`
: Input. A pointer to a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`, to specify the buyer whose suspension is to be made.

`domain`
: Input. The domain to which the buyer is denied access. If this parameter is null (“”), the buyer will be denied from all domains.

Return Values

The `SuspendBuyerResource` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJBuyerService::UpdateCustomFields

Name

`UpdateCustomFields` – Update a buyer's custom fields.

Synopsis

```
virtual HRESULT UpdateCustomFields(  const VWSJContext* ctxPtr,
                            VWSJBuyer* buyer,
                            const TExtensionData& exData,
                            const Bool getLatest) = 0;
```

Description

This method updates a buyer's custom fields.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyer`
: Input. A pointer to a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`, to specify the buyer whose custom fields are to be updated.

`exData`
: Input. The criteria to set.

`getLatest`
: Input. A flag to return, in the buyer object, the most recent extension data for this buyer.

Return Values

The `UpdateCustomfields` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJBuyerService::UpdatePasswordInTx

Name

`UpdatePasswordInTx` – Update a buyer's password in Transact database.

Synopsis

```
virtual HRESULT UpdatePasswordInTx(  const VWSJContext* ctxPtr,
                                const VWSJBuyer* buyer) = 0;
```

Description

This method update's a buyer's password.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyer`
: Input. A pointer to a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`, to specify the buyer whose password is to be updated.

Return Values

The `UpdatePasswordInTx` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJBuyerService::VerifyChallenge

Name

`VerifyChallenge` – Verify the challenge string provided by buyer.

Synopsis

```
virtual HRESULT VerifyChallenge(
                  const VWSJContext* ctxPtr,
                  const VWSJBuyer* buyer,
                  const TString& question,
                  const TString& answer) = 0;
```

Description

This method specifies a search string and then verifies whether it matches the challenge string stored in the database for the specified buyer.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`buyer`
: Input. A pointer a VWSJBuyer constant, i.e., `const VWSJBuyer* buyer`, to specify the buyer for whom the search string is tested.

`question`
: Input. The search string to be verified.

`answer`
: Output.

Return Values

The `VerifyChallenge` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJContext Class

Name

`VWSJContext` – The WSJ context class.

Description

Methods in this class allow login to the Transact database, generation of GUIDs, logging of mesages, and access to (and changing of) properties such as user name and logging, store ID, address information, order number, etc.

Public Methods

```
AppLogin
GenerateGUID
GenerateGUID (#2)
Get Data
Login
LogMessage
Set Data

virtual Bool IsValid() const = 0;
```

Related Classes

```
VWSJ. . .
```

VWSJContext::AppLogin

Name

`AppLogin` – Log an application into the Transact database.

Synopsis

```
virtual Bool AppLogin (TDB Session* dbSessionPtr) =0;
```

Description

Log an application into the Transact database. (You need to have used the `SetKeyStoreID` method on this object first.)

Parameters

`dbSessPtr`
: Input parameter. A pointer to a DBSession object.

Return Values

The `AppLogin` method returns:

`TRUE`
: If the method successfully logged in the application.

`FALSE`
: If the method could not successfully log in.

Example

```
TWSJFoundationFactory factory;
VWSJContext *wsjCtx = factory.MakeVWSJContext();
wsjCtx->SetKeyStoreID(smartData.keyStoreID);
if (! wsjCtx->AppLogin()) {
    wsjCtx->LogMessage(__LINE__,
        TString(__FILE__)+TString(":")+funcname+TString(":
        ERROR: Failed to login to the application"));
    return E_FAIL;
}
```

VWSJContext::Cleanup

Name

`Cleanup` – Cleanup Transact database.

Synopsis

```
virtual void Cleanup () =0;
```

Description

`Clean` checks the validity of the database objects and tests for an open database transaction.

If a transaction is open, `Cleanup` commits the transaction and returns S_OK.

It closes the DB connection if the connection is *not* controlled by a UI application.

Return Values

The `Cleanup` method returns:

`S_OK`
: If a transaction was open and `Cleanup` was successfuly in committing it.

Example

VWSJContext::GenerateGUID

Name

`GenerateGUID` – Generate a GUID.

Synopsis

```
virtual TString GenerateGUID (const TString &tableName) =0;
```

Description

Generate a GUID for the row to be entered into the table specified.

Parameters

`tableName`
: Input parameter. A pointer to the table name to be added to the GUID.

Return Values

The `GenerateGUID` method returns:

`A GUID`
: If the method successfully added the table name.

`NULL TString`
: If the method was unsuccessful.

VWSJContext::GenerateGUID (#2)

Name

`GenerateGUID` – Generate a GUID.

Synopsis

```
virtual TString GenerateGUID (
                        const TString &tableName,
                        const long& order) =0;
```

Description

Generate a GUID for the specified row and order number to be entered into the table specified.

Parameters

`tableName`
: Input parameter. A pointer to the table name to be added to the GUID.

`order`
: Input parameter. A pointer to the order number to be added to the GUID.

Return Values

The `GenerateGUID` method returns:

`A GUID`
: If the method successfully added the table name and order number.

`NULL TString`
: If the method was unsuccessful.

VWSJContext::Get*Data*

Name

`Get*Data*` – Access a store's key ID or GUID.

Synopsis

```
virtual type GetValue () const =0;
```

Description

This group of methods accesses a store's ID and key ID values in the WSJ database.

Parameters

`Value`
The data item being requested

Table 23: Accessor Methods for VWSJContext Class

Type	Method	Description and Notes
`TString`	`GetKeyStoreID`	The store's key ID
`GUIDString`	`GetLoginID`	The buyer's GUID
`GUIDString`	`GetStoreGUID`	The store's GUID
`TString`	`GetDefaultFirstName`	The buyer's first name
`TString`	`GetDefaultLastName`	The buyer's last name
`TString`	`GetDefaultAddress1`	The first line of buyer's address
`TString`	`GetDefaultAddress2`	The second line of buyer's address
`TString`	`GetDefaultCity`	The buyer's city
`TString`	`GetDefaultState`	The buyer's state
`TString`	`GetDefaultPostalCode`	The buyer's postal coard (e.g., zip code)
`TString&`	`GetDefaultCtryCode`	The buyer's country code
`TString`	`GetDefaultPhone`	The buyer's phone number
`TString`	`GetDefaultCatalogURL`	The store's catalog URL
`TString`	`GetDefaultFulfillment URL`	The store's fulfillment URL

Table 23: Accessor Methods for VWSJContext Class

Type	Method	Description and Notes
`TString`	`GetDefaultTicket Format`	The store's default ticket format

Notes

In general, the buyer logs in to a particular store when he logs into a context. But the WSJ only has one store. See *Transact 4 Utility Classes API*.

VWSJContext::Login

Name

`Login` – Log into the Transact database.

Synopsis

```
virtual Bool Login () =0;
```

Description

Log into the Transact database. (You need to have used the `SetKeyStoreID` method on this object first.)

Parameters

None

Return Values

The `Login` method returns:

`TRUE`
: If the method successfully logged in.

`FALSE`
: If the method could not successfully log in.

Example

```
TString user("transactmerchant");
TString userPassword("challenge");
int storeID = 1308;
context->SetLoginUserName(user);
context->SetLoginUserPassword(userPassword);
context->SetKeyStoreID(TString::IntToString(storeID));
Bool lResult = context->Login();
if (! lResult) {
    return FALSE;
}
```

VWSJContext::LogMessage

Name

`LogMessage` – Write to the log server as set up in the registry.

Synopsis

```
virtual void LogMessage (
                        const int& lineno,
                        const TString& msg) =0;
```

Description

Write a message to the log file specified in the registry.

Parameters

`lineno`
: Input parameter. A pointer to the line in the file to receive the message.

`msg`
: Input parameter. A pointer to the message to be added to the log file.

Return Values

None

Example

```
smartData.keyStoreID = TString::IntToString (storeID);
smartData.principalID = reqCtx.fCurrentUser.GetID ();

TWSJFoundationFactory factory;
VWSJContext *wsjCtx = factory.MakeVWSJContext();
wsjCtx->SetKeyStoreID(smartData.keyStoreID);
if (! wsjCtx->AppLogin()) {
    wsjCtx->LogMessage(__LINE__,
          TString(__FILE__)+TString(":")+funcname+
          TString(": ERROR: Failed to login to the application"));
    return E_FAIL;
}
```

VWSJContext::Set*Data*

Name

`Set`*`Data`* – Specify the value of the specified data item.

Synopsis

```
virtual void Set Value (type Value) =0;
```

Description

This group of methods sets mostly default buyer contact data values into the Transact database.

Parameters

`Value`
The data item being set.

Table 24: Mutator Methods for VWSJContext Class

Method	Data Parameter	Description
`SetCurrentOrderNumber`	`const long& in`	Not currently employed in system
`SetDefaultAddress1`	`const TString& in`	Default buyer contact data
`SetDefaultAddress2`	`const TString& in`	Default buyer contact data
`SetDefaultCity`	`const TString& in`	Default buyer contact data
`SetDefaultCtryCode`	`const TString& in`	Default buyer contact data
`SetDefaultPhone`	`const TString& in`	Default buyer contact data
`SetDefaultPostalCode`	`const TString& in`	Default buyer contact data
`SetDefaultState`	`const TString& in`	Default buyer contact data
`SetKeyStoreID`	`const TString& in`	The store's key ID
`SetLoginUserName`	`const TString& in`	Login name for Transact database
`SetLoginUserPassword`	`const TString& in`	Password for Transact database

Return Values

None.

VWSJDomain Class

Name

`VWSJDomain` – The WSJ resource or domain.

Description

Methods in this class access and change properties such as a domain's name, ID, and URL.

Public Methods

`Get`*`Data`*
`Set`*`Data`*

```
virtual Bool IsValid() const = 0;
```

Related Classes

`VWSJProductService`

`VWSJDomainVector`

VWSJDomain::Get*Data*

Name

`Get`*`Data`* – Access data related to a domain or resource.

Synopsis

```
virtual type GetValue () const =0;
```

Description

This group of methods accesses data values in the WSJ database pertaining to a domain.

Parameters

`Value`
: The data item being requested

Table 25: Accessor Methods for VWSJDomain Class

Type	Method	Description and Notes
`GUIDString`	`GetID`	GUID for the domain
`TString`	`GetFulfillmentURL`	The domain URL
`TString`	`GetName`	The domain name

Example

```
TPmtInfo info = StringToInfo (qsData);
. . .
info.Insert (kResourceName, domain->GetName());
info.Insert (kResourceURL, domain->GetFulfillmentURL());
```

VWSJDomain::Set*Data*

Name

`Set*Data*` – Specify data for a domain.

Synopsis

```
virtual void Set*Value* (*type Value*) =0;
```

Description

This group of methods sets data values into the WSJ database for domains.

Parameters

`Value`
The data item being set.

Table 26: Mutator Methods for VWSJDomain Class

Method	Data Parameter	Description
`SetFulfillmentURL`	`const TString& URL`	URL of the fulfillment domain
`SetID`	`const GUIDString& id`	The GUID for the domain
`SetName`	`const TString& name`	The name of the domain

Example

```
if (domain->GetName () == resourceName) {
    domain->SetFulfillmentURL (resourceURL);
    if (FAILED (domainService->UpdateDomain (wsjCtx, domain))) {
        return E_FAIL;
    }
} else {
    domain->SetID ("");
    domain->SetName (resourceName);
    domain->SetFulfillmentURL (resourceURL);
    if (FAILED (domainService->CreateDomain (wsjCtx, domain))) {
        return E_FAIL;
    }
}
```

VWSJDomainService Class

Name

`VWSJDomainService` – The service class for WSJ domains.

Description

Methods in this class create domains in the database; look them up by ID or name; update them; and get their names.

Public Methods

`CreateDomain`
`GetAllDomains`
`LookUpBy`*`Data`*
`UpdateDomain`

Related Classes

`VWSJDomain`

VWSJDomainService::CreateDomain

Name

`CreateDomain` – Create a domain object.

Synopsis

```
virtual HRESULT CreateDomain(
                  const VWSJContext* ctxPtr,
                  VWSJDomain* domain) = 0;
```

Description

This method creates a domain in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`domain`
: Input and output. A pointer to a VWSJDomain object to specify the domain to be created.

Return Values

The `CreateDomain` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

Example

```
if (domain->GetName () == resourceName) {
    domain->SetFulfillmentURL (resourceURL);
    if (FAILED (domainService->UpdateDomain (wsjCtx, domain))) {
        return E_FAIL;
    }
} else {
    domain->SetID ("");
    domain->SetName (resourceName);
    domain->SetFulfillmentURL (resourceURL);
    if (FAILED (domainService->CreateDomain (wsjCtx, domain))) {
        return E_FAIL;
    }
}
```

VWSJDomainService::GetAllDomains

Name

`GetAllDomains` – Get all domain names.

Synopsis

```
virtual HRESULT GetAllDomains(
                  const VWSJContext* ctxPtr,
                  TWSJDomainVector& domains) = 0;
```

Description

This method retrieves a vector of GUIDs of all the domains in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`domains`
: Output. A vector of the domain names.

Return Values

The `GetAllDomains` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJDomainService::LookUpBy*Data*

Name

`LookUpBy`*`Data`* – Access the value of the specified data.

Synopsis

```
virtual HRESULT LookUpBy DomainProperty ( . . . ) =0;
```

Description

This group of methods accesses domain objects by name and ID in the WSJ database.

Parameters

Several, and varied. See Table 27, "LookUp Accessor Methods for VWSJDomainService Class," for their usage.

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by nearly all Lookup methods for the VWSJDomainService class. It specifies the object containing the operational context required to perform database operations.

`ID`
: Input. A pointer to a GUIDString containing the ID of the domain for which a pointer is to be returned.

`domainName`
: Input. A TString containing the name of the domain for which a pointer is to be returned.

`domain`
: Output. A pointer to a VWSJDomain object that will contain the pointer to the domain found by the LookUp methods.

Table 27: LookUp Accessor Methods for VWSJDomainService Class

Method	Parameters	Description and Notes
`LookUpByName`	`const VWSJContext* ctxPtr, const TString& domainName, VWSJDomain* domain`	Return a domain given its name
`LookUpByID`	`const VWSJContext* ctxPtr, const GUIDString& id, VWSJDomain* domain`	Return a domain given its GUID

Status Returns

All the LookUp methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

```
if (! selectedResource.IsNull()) {
    hr = domainService->LookUpByID (wsjCtx, selectedResource, domain);
    if (FAILED (hr)) {
       return E_FAIL;
    }
```

VWSJDomainService::UpdateDomain

Name

`UpdateDomain` – Update a domain object.

Synopsis

```
virtual HRESULT UpdateDomain(
                        const VWSJContext* ctxPtr,
                        VWSJDomain*  domain) = 0;
```

Description

This method updates a domain object in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`domain`
: Input. The domain object to update.

Return Values

The `UpdateDomain` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJEncryptService Class

Name

`VWSJEncryptService` – A class for string encryptions.

Description

The method in this class decrypts an encrypted string. Effectively, this publicizes a method to wrap private Transact encrytion routines.

Public Methods

`GetDecryptStringFromEncryptString`

VWSJEncryptService::GetDecryptStringFromEncryptString

Name

`GetDecryptStringFromEncryptString` – Return a decrypted value

Synopses

```
virtual HRESULT GetDecryptStringFromEncryptString(
                            const TString &in, TString& out) const = 0;
```

Description

This method returns a decrypted string from an encrypted string.

Parameters

`in`
: Input parameter. The encrypted string.

`out`
: Output parameter. The decrypted string.

Return Values

None.

VWSJGroup Class

Name

`VWSJGroup` – The WSJ group.

Description

Methods in this class access and change properties such as a the name of the subscriber's group, its description, address, phone, and maximum number of subscribers allowed, etc. This class is used for access control.

Enumerated Data

```
enum EGroupType {
    kGroupTypeUnknown,
    kGroupTypeFree,
    kGroupTypeComp
};
```

Table 28: Meanings of EGroupType Groups

Value of EGroupType	Group Subscription Type
`kGroupTypeComp`	Complimentary
`kGroupTypeFree`	Free
`kGroupTypeUnknown`	Unknown

Public Methods

```
AddResources
Disable
Enable
Get Data
IsEnabled
IsSubscriber
Set Data
VerifyRegistrationSecret

virtual Bool IsValid() const = 0;
```

Related Classes

`VWSJGroupService`

`VWSJGroupVector`

VWSJGroup::AddResources

Name

`AddResources` – Add resources to a group.

Synopsis

```
virtual HRESULT AddResources (
           const TGUIDStringVector& resourceIDs) =0;
```

Description

Add one or more resources to a group.

Parameters

`resourceIDs`
: Input parameter. A vector of resource GUIDs.

Return Values

`HRESULT . .`

Example

```
value = "";
info.Lookup (kResourceAccess, value);
RWTValOrderedVector<RWCString> frags;
SplitString (value.AsString().ToUTF(), ',', frags);
TGUIDStringVector resourceIDs;
for (int i=0; i<frags.length(); i++) {
   resourceIDs.Append (frags[i].data());
}
groupPtr->AddResources (resourceIDs);
```

VWSJGroup::Get*Data* (and Related Accessors)

Names

`Get`*`Data`* – Access the value of the specified data related to the group.

`IsEnabled` – Access the enabled status of the group.

`IsSubscriber` – Access the subscriber status of a user ID.

`VerifyRegistrationSecret` – Check a registration "secret."

Synopses

```
virtual ReturnType GetValue () const =0;

virtual Bool IsEnabled () const =0;

virtual Bool IsSubscriber (const GUIDString& userID) =0;

virtual Bool VerifyRegistrationSecret (const TString& secret) =0;
```

Description

This group of methods accesses data values in the WSJ database regarding group information, such as admin login name, CSR contact, buyer's group contact and secret, etc.

Parameters

`Value`
: The data item being requested

`userID`
: The user ID to test for being a subscriber

`secret`
: The registration secret to verify

Table 29: Accessor Methods for VWSJGroup Class

Return Type	Method	Description and Notes
`TString`	`GetAdminLoginName`	Login name of administrator for group
`TString`	`GetDescription`	Group's description
`TString`	`GetDisabledMessage`	Text to display when subscriber can't access group's content (see IsEnabled method below)
`TString`	`GetDJRepName`	Name of the DJ rep to the group
`TString`	`GetDJRepPhone`	The DJ rep's phone number
`TString`	`GetGroupContactName`	Contact at group's location for WSJIE
`TString`	`GetGroupContactPhone`	Contact's phone number
`GUIDString`	`GetID`	GUID for this group
`long`	`GetMaxSubscribers`	Maximum number of subscribers allowed
`TString`	`GetName`	Name of the group
`TString`	`GetNotes`	Free-form notes for group information
`TAddress`	`GetOrgAddress`	Group's street, city, state, zip, country
`TGUIDStringVector`	`GetResources`	Vector of GUIDs for which products are available in this group
`TString`	`GetSourceKey () = 0;`	Key of WSJ store
`GUIDString`	`GetStoreID`	GUID for the store for which this group was created
`TGUIDStringVector`	`GetSubscribers`	A potentially really long list
`EGroupType`	`GetType`	Complimentary, free, or paid? See Table 17
`Bool`	`IsEnabled`	Is group enabled or disabled?
`Bool`	`IsSubscriber (const GUIDString& userID) =0;`	Is GUID that of a subscriber in this group?
`Bool`	`VerifyRegistrationSecret (const TString& secret) =0;`	Does the string passed in match the secret for the group?

VWSJGroup::RemoveResource, RemoveResources

Name

`RemoveResource` – Remove a resource from a group.

`RemoveResources` – Remove multiple resources from a group.

Synopses

```
virtual void RemoveResource (
          const TString& resourceID) =0;

virtual void RemoveResources (
          const TGUIDStringVector& resourceIDVector) =0;
```

Description

Remove one or more resources from a group.

Parameters

`resourceID`
: Input parameter. A single resource ID.

`resourceIDVector`
: Input parameter. A vector of resource IDs.

Return Values

None.

VWSJGroup::Set*Data* (and Related Mutators)

Names

`Set`*`Data`* – Specify the value of a group data item.

`Disable` – Disable a specified user ID.

`Enable` – Enable a specified user by name.

Synopses

```
virtual void Set Value (type Value) =0;

virtual void Disable (type Value) =0;

virtual void Enable (type Value) =0;
```

Description

This group of methods sets data values into the WSJ database concerning group information, such as admin login name, CSR contact, buyer's group contact and secret, etc

Parameters

`Value`

The data item being set.

Table 30: Mutator Methods for VWSJGroup Class

Method	Data Parameter	Description
`Disable`	`const GUIDString& id`	Disable group with this GUID
`Enable`	`const TString& name`	Enable group with this name
`SetAdminLoginName`	`const TString& name`	Login name for logging into database for this group
`SetCreatorID`	`const GUIDString& creatorID`	Login name for creator of group
`SetDescription`	`const TString& description`	Description of group
`SetDisabledMessage`	`const TString& message`	Text to display when group is disabled

Table 30: Mutator Methods for VWSJGroup Class

Method	Data Parameter	Description
SetDJRepName	const TString& name	The DJ rep to this group
SetDJRepPhone	const TString& phone	The rep's phone number
SetGroupContactName	const TString& name	Contact at the group's location
SetGroupContactPhone	const TString& phone	Contact's phone number
SetID	const GUIDString& id	GUID for this record
SetMaxSubscribers	const long maxSubs	Limit for number of subscribers in this group
SetName	const TString& name	Name of group
SetNotes	const TString& notes	Free-form textual note
SetOrgAddress	const TAddress& address	Group's organizational address
SetRegistrationSecret	const TString& secret	Secret the subscriber has to provide to register for access to the group
SetSourceKey	const TString& sourceKey	Key of WSJ store
SetStoreID	const TString& storeID	ID for store from which group buys
SetTicketExpiration	const TDate& expiration	When the ticket expires for access to the group (the ticket is Transact-issued data that lets the subscriber access content)
SetType	const EGroupType& type	Subscription type: complimentary, free, or paid; see Table 17

Return Values

None.

Example

```
value = "";
info.Lookup (kGroupType, value);
if (value.AsString() == kCOMP) {
    groupPtr->SetType (VWSJGroup::kGroupTypeComp);
} else if (value.AsString() == kFree) {
    groupPtr->SetType (VWSJGroup::kGroupTypeFree);
}

value = "";
info.Lookup (kMaxSubscribers, value);
groupPtr->SetMaxSubscribers (value.AsInt());
```

VWSJGroupService Class

Name

`VWSJGroupService` – The service class for the WSJ group class.

Description

Methods in this class create groups in the database; look them up by ID or name; update them; and get their names.

Public Methods

```
CreateGroup
GetGroupsBySubscriber
LookUpData
UpdateGroup

static Bool Test();
```

Related Classes

`VWSJDomain`

VWSJGroupService::CreateGroup

Name

`CreateGroup` – Create a group object.

Synopsis

```
virtual HRESULT CreateGroup(
                  const VWSJContext* ctxPtr,
                  VWSJGroup* group) = 0;
```

Description

This method creates a group in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`group`
: Input and output. A pointer to a VWSJGroup object to specify the group to be created.

Return Values

The `CreateGroup` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJGroupService::GetGroupsBySubscriber

Name

`GetGroupsBySubscriber` – Get all groups for a subscriber.

Synopsis

```
virtual HRESULT GetGroupsBySubscriber(
                    const VWSJContext* ctxPtr,
                    const GUIDString& subscriberID,
                    TGUIDStringVector& groupIDs) = 0;
```

Description

This method retrieves the names of all the groups for a specific subscriber in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`subscriberID`
: Input. The subscriber's ID.

`groupIDs`
: Output. A vector of the group names.

Return Values

The `GetGroupsBySubscriber` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJGroupService::LookUp*Data*

Name

`LookUp`*`ByData`* – Access the group according to ID or name.

Synopsis

```
virtual HRESULT LookUpGroupByProperty ( . . . ) =0;
```

Description

This group of methods accesses group objects by name or ID in the WSJ database.

Parameters

Several, and varied. See Table 31, "LookUp Accessor Methods for VWSJGroupService Class," for their usage.

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Lookup methods for the VWSJGroupService class. It specifies the object containing the operational context required to perform database operations.

`ID`
: Input . A pointer to a GUIDString containing the ID of the group for which a pointer is to be returned.

`name`
: Input . A TString containing the name of the group for which a pointer is to be returned.

`group`
: Output . A pointer to a VWSJGroup object that will contain the pointer to the group found by the LookUp methods.

Table 31: LookUp Accessor Methods for VWSJGroupService Class

Method	Parameters	Description and Notes
`LookUpByName`	`const VWSJContext* ctxPtr, const TString& groupName, VWSJGroup* group`	Return a group object given a name
`LookUpByID`	`const VWSJContext* ctxPtr, const GUIDString& id, VWSJGroup* group`	Return a group object given a GUID

Status Returns

All the LookUp methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

```
for (int i=0; i<prodGuidVector.length(); i++) {
    hr = prodService->LookUpByName (wsjCtx, prodGuidVector[i],\
                                    product);
    if (FAILED (hr)) {
        TOMKTStringVector params;
        params.Append (prodGuidVector[i]);
        error.AddErrorItem (TOMKTErrorItem ("WSJuser", kSWSJ, kError,\
                                    kSWSJ_ProductNotFound, params));
        return E_FAIL;
}
    prodName2Guid.Insert (prodGuidVector[i], product->GetProductID ());
}
```

VWSJGroupService::UpdateGroup

Name

`UpdateGroup` – Update a group object.

Synopsis

```
virtual HRESULT UpdateGroup(
                    const VWSJContext* ctxPtr,
                    VWSJGroup* group) = 0;
```

Description

This method updates a group object in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`group`
: Input. The group object to update.

Return Values

The `UpdateGroup` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJOffer Class

Name

`VWSJOffer` – The WSJ offer class.

Description

Methods in this class access and change properties related to offers being made to a buyer. These include the adding products, offer ID, name, description, number of days, and miscellaneous data; as well as other agreement registration, filename, date last modified, and creator ID.

Public Methods

```
AddProducts
Get Data
GetProducts
Set Data

virtual Bool IsValid() const = 0;
```

Related Classes

`VWSJOfferService`

VWSJOffer::AddProducts, GetProducts

Names

`AddProducts` – Add products to an offer.

`GetProducts` – Get a vector of products in an offer.

Synopses

```
virtual void AddProducts (
                const TGUIDStringVector& productGUIDs) =0;

virtual void GetProducts (
                TGUIDStringVector& productGUIDs) =0;
```

Description

Add products to an offer; get the list of products in an offer.

Parameters

`productGUIDs`

A vector of product GUIDs to be added to an offer or to receive a list of the GUIDs in an offer.

Return Values

None.

VWSJOffer::Get*Data*

Name

`Get`*`Data`* – Access data related to offers.

Synopsis

```
virtual ReturnType GetValue () const =0;
```

Description

This group of methods accesses data values in the WSJ database relating to offers, e.g., time period, notes, name, registration web page, etc

Parameters

`Value`

The data item being requested

Table 32: Accessor Methods for VWSJOffer Class

Return Type	Method	Description and Notes
`TString`	`GetAgreementFileName`	Name of file containing text agreed to by buyer; has to be in Transact's text directory
`TDate`	`GetCreateTime`	Time this offer record was created
`GUIDString`	`GetCreatorID`	GUID of the admin (creator) of this offer
`TString`	`GetDescription`	Description of offer
`TDate`	`GetEndDate`	When the offer ends
`int`	`GetIFPDays`	How many Initial Free Period days to allow
`TDate`	`GetLastModifiedTime`	When offer was last modified
`TString`	`GetNotes`	Free-form notes about the offer
`GUIDString`	`GetOfferID`	GUID of this offer
`TString`	`GetOfferName`	Name of offer
`TString`	`GetRegistrationPage`	The name of the file used for registration; the file has to be in Transact's text directory
`TString`	`GetSourceKey`	Key of WSJ store
`TDate`	`GetStartDate`	When offer is first offered

Examples

```
VWSJOffer *offer = factory.MakeVWSJOffer ();
for (int i=0; i<offerIDs.Length(); i++) {
          if (FAILED (offerService->LookUpByID (wsjCtx,offerIDs[i],offer))){
              return E_FAIL;
          }
smartData.curOfferNamesLabels.insert (offer->GetOfferName());
smartData.curOfferNamesValues.insert (offer->GetOfferID());
}
```

The following example looks up an offer ID.

```
TSmartPtr<VWSJOffer> offerPtr = factory.MakeVWSJOffer();
if(FAILED(offerPtr->IsValid())) {
    cout << "main: Error: offer is invalid" << endl;
    return FALSE;
}
TSmartPtr<VWSJOfferService> pOfferService =  factory.MakeVWSJOfferService();
if(FAILED(pOfferService->LookUpByName(context, gOfferName, offerPtr.get())))
{
    cout << "main: Error: could not find offer name="<< gOfferName << endl;
    return FALSE;
}
offerID = offerPtr->GetOfferID();
```

VWSJOffer::Set*Data*

Name

`Set`*`Data`* – Specify the value of an offer-related data item.

Synopsis

```
virtual void Set Value  (type Value) =0;
```

Description

This group of methods sets data values into the WSJ database for offers, for example, their creation date and author, associated agreement file, store, registration URL, etc.

Parameters

`Value`

The data item being set.

Table 33: Mutator Methods for VWSJOffer Class

Method	Data Parameter	Description
`SetAgreement\ FileName`	`const TString& AgreementFileName`	Name of file containing text agreed to by buyer; has to be in Transact's text directory
`SetCreateTime`	`const TDate& CreateTime`	Time this offer record was created
`SetCreatorID`	`const GUIDString& CreatorID`	GUID of the admin (creator) of this offer
`SetDescription`	`const TString& Description`	Type of offer
`SetEndDate`	`const TDate& EndDate`	When the offer ends
`SetIFPDays`	`const int& IFPDays`	How many Initial Free Period days to allow
`SetLastModifiedTime`	`const TDate& LastModifiedTime`	When offer was last modified
`SetNotes`	`const TString& Notes`	Free-form notes about the offer
`SetOfferID`	`const GUIDString& OfferID`	GUID of this offer

Table 33: Mutator Methods for VWSJOffer Class

Method	Data Parameter	Description
`SetOfferName`	`const TString& offerName`	Name of offer
`SetRegistrationPage`	`const TString& RegistrationPage`	The name of the file used for registration; the file has to be in Transact's text directory
`SetSourceKey`	`const TString& SourceKey`	Key of WSJ store
`SetStartDate`	`const TDate& StartDate`	When offer is first offered

Return Values

None.

Example

```
GUIDString offerID = GenerateGUIDStr();
offer->SetOfferID(offerID);
offer->SetCreatorID(smartData.principalID);
offer->SetDescription (formData.GetValue (kOfferDesc));
offer->SetSourceKey (formData.GetValue (kSourceKey));
offer->SetIFPDays (
atoi ((formData.GetValue (kLengthOfIFP)).ToUTFBytes()));
```

VWSJOfferService Class

Name

`VWSJOfferService` – The service class for the WSJ offer class.

Description

Methods in this class create offers in the database; look them up by ID or name; update them; and add products to them.

Public Methods

```
AddProduct
CreateOffer
GetAllOffers
LookUpByID
LookUpByName
UpdateOffer
```

Related Classes

`VWSJOffer`

VWSJOfferService::AddProduct

Name

`AddProduct` – Add a product to an offer.

Synopsis

```
virtual HRESULT AddProduct(
                  const VWSJContext* ctxPtr,
                  const GUIDString& offerID,
                  const GUIDString& productID) = 0;
```

Description

Add a product to an offer.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Lookup methods for the VWSJOfferService class. It specifies the object containing the operational context required to perform database operations.

`offerID`
: Input . A pointer to a GUIDString containing the ID of the offer to which a product is to be added.

`productID`
: Input . A pointer to a GUIDString containing the ID of the product which is to be added.

Return Values

The `AddProduct` method returns:

`S_OK`
: If the method successfully added or obtained the product(s).

`E_FAIL`
: If the method was not successful.

<DB Errors>
: If the method resulted in database errors.

Example

```
for (int i=0; i<productGuids.Count(); i++) {
    TPmtInfo::Key key;
    TPmtInfo::Value value;
    iterator.Next (key, value);
    hr = offerService->AddProduct (wsjCtx,
                         offerID,
                         value.AsString());
    if (FAILED (hr)) {
        return E_FAIL;
    }
}
```

VWSJOfferService::CreateOffer

Name

`CreateOffer` – Create an offer object.

Synopsis

```
virtual HRESULT CreateOffer(
                  const VWSJContext* ctxPtr,
                  VWSJOffer* offer) = 0;
```

Description

This method creates an offer in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`offer`
: Input and output. A pointer to the VWSJOffer object that was created.

Return Values

The `CreateOffer` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJOfferService::GetAllOffers

Name

`GetAllOffers` – Get all offers.

Synopsis

```
virtual HRESULT GetAllOffers (
                  const VWSJContext* ctxPtr,
                  TWSJOfferVector& offervec) = 0;
```

Description

Get a vector of all the WSJIE offers.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Lookup methods for the VWSJOfferService class. It specifies the object containing the operational context required to perform database operations.

`offervec`
: Output . A vector of offer IDs returned by the method.

Return Values

The `GetAllOffers` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJOfferService::LookUpByID

Name

`LookUpByID` – Access the offer according to ID.

Synopsis

```
virtual HRESULT LookUpByID(
                  const VWSJContext* ctxPtr,
                  const GUIDString& id,
                  VWSJOffer* offer) = 0;
```

Description

This method accesses offer objects in the WSJ database.

Parameters

`ctxPtr`

Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Lookup methods for the VWSJOfferService class. It specifies the object containing the operational context required to perform database operations.

`id`

Input . A pointer to a GUIDString containing the ID of the offer for which a pointer is to be returned.

`offer`

Output . A pointer to a VWSJOffer object that will contain the pointer to the offer found by the LookUp method.

Return Values

`S_OK`

If the method succeeded.

`E_FAIL`

If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

```
TString selectedOffer = formData.GetValue (kCurOffers);
hr = offerService->LookUpByID (wsjCtx, selectedOffer, offer);
if (FAILED (hr)) {
    return E_FAIL;
}
```

VWSJOfferService::LookUpByName

Name

`LookUpByName` – Access the offer according to name.

Synopsis

```
virtual HRESULT LookUpByName(
                  const VWSJContext* ctxPtr,
                  const TString& name,
                  VWSJOffer* offer) = 0;
```

Description

This method accesses offer objects in the WSJ database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. It specifies the object containing the operational context required to perform database operations.

`name`
: Input . A pointer to a GUIDString containing the name of the offer for which a pointer is to be returned.

`offer`
: Output . A pointer to a VWSJOffer object that will contain the pointer to the offer found by the LookUp method.

Status Returns

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

VWSJOfferService::UpdateOffer

Name

`UpdateOffer` – Update an offer object.

Synopsis

```
virtual HRESULT UpdateOffer(
            const VWSJContext* ctxPtr,
            VWSJOffer* offer) = 0;
```

Description

This method updates an offer object in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`offer`
: Input. The offer object to update.

Return Values

The `UpdateOffer` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJPmtAccount Class

Name

`VWSJPmtAccount` – The WSJ payment account class.

Description

Methods in this class access and change properties related to a buyer's credit card, e.g., expiration date, brand, name on card, number, and type. (That is, to the buyer's means of payment for this account.)

Public Methods

`Get`*`Data`*
`Set`*`Data`*

```
virtual Bool IsValid() const = 0;
```

Related Classes

`VWSJPmtAccountService`

VWSJPmtAccount::Get*Data*

Name

`Get`*`Data`* – Access a value in the payment account.

Synopsis

```
virtual type GetValue () const =0;
```

Description

This group of methods accesses data values in the WSJ database related to how this account is paid.

Parameters

`Value`

The data item being requested

Table 34: Accessor Methods for VWSJPmtAcount Class

Return Type	Method	Description and Notes*
`GUIDString`	`GetBrandID`	GUID of this brand
`TString`	`GetCCNumber`	Number on credit card
`TString`	`GetCCType`	VI, MC, DC, DI, AM
`TString`	`GetExpirationDate`	When the card expires
`GUIDString`	`GetID`	The GUID for this record
`HRESULT`	`GetLast4CCDigits( TString& number) = 0;`	Last four digits of credit card number
`TString`	`GetNameOnCard`	Buyer's name on the card

* See *Payment API*

VWSJPmtAccount::Set*Data*

Name

`Set`*`Data`* – Specify the value of a payment account data item

Synopsis

```
virtual { void | HRESULT } Set Value (type Value) =0;
```

Description

This group of methods sets data values into the WSJ database relating to how this account is paid.

Parameters

`Value`

The data item being set.

Table 35: Mutator Methods for VWSJPmtAccount Class

Method (and Return Type if not Void)	Data Parameter	Description*
`SetBrandID`	`const GUIDString& brand`	GUID of this brand
`SetCCNumber`	`const TString& number`	Credit card number
`SetCCType`	`const TString& type`	VI, MC, DC, DI, AM
`SetExpirationDate (HRESULT)`	`const TString& expire`	When the card expires
`SetNameOnCard`	`const TString& name`	Buyer's name on card

* See *Payment API*

VWSJPmtAccountService Class

Name

`VWSJPmtAccountService` – The service class for the WSJ payment account class.

Description

Methods in this class look up payment accounts in the database by ID. A *payment account* is generally a credit card account.

Public Methods

`LookUpByID`

Related Classes

`VWSJPmtAccount`

VWSJPmtAccountService::LookUpByID

Name

`LookUpByID` – Access the payment account according to ID.

Synopsis

```
virtual HRESULT LookUpByID(
                  const VWSJContext* ctxPtr,
                  const GUIDString& pmtAccountId,
                  VWSJPmtAccount* PmtAccountPtr) = 0;
```

Description

This method looks up all information on the requested VWSJPmtAccount object based on the ID passed in and returns a full VWSJPmtAccount object.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant to specify the object containing the context required to perform database operations.

`pmtAccountId`
: Input . A pointer to the GUID for the payment account for which a pointer is to be returned.

`PmtAccount`
: Input and Output . A pointer to a VWSJPmtAccount object that will contain the pointer to the PmtAccount found by the LookUp method.

Status Returns

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

VWSJProduct Class

Name

`VWSJProduct` – The WSJ product class.

Description

Methods in this class add, remove, access, and change a product's name, cost, and related resources; whether it is available, etc.

Enumerated Data

```
enum EProductDuration {
    kYearly,
    kMonthly
};
```

Table 36: Meanings of EProductDuration Types

Value of EproductDuration	How Often Billed
`kMonthly`	Monthly
`kYearly`	Yearly

Public Methods

```
AddResource
Get<i>Data</i>
IsAvailable
IsBillable
RemoveResource
Set<i>Data</i>

virtual Bool IsValid() const = 0;
```

VWSJProduct::Add*Resource*

Name

`AddResource, AddResources` – Add a single resource or a set of resources.

Synopsis

```
virtual HRESULT AddResource (
        const GUIDString& resourceID) =0;

virtual HRESULT AddResources (
        const TGUIDStringVector& resourceIDs) =0;
```

Description

The `Add` methods add one or more resources to a product.

Parameters

`resourceID, resourceIDs`
: Input parameter. A single resource ID, or a vector of resource IDs.

Return Values

`HRESULT`

Related Methods

`VWSJProduct::RemoveResource`

`VWSJProduct::RemoveResources`

Example

```
TGUIDStringVector resourceIDs;
resourceIDs.Append(domainPtr->GetID());
productPtr->AddResources(resourceIDs);
```

VWSJProduct::Get*Data* (and Related Accessors)

Names

`Get`*`Data`* – Access a data item related to a product.

`IsAvailable` – Check to see if a product is available.

`IsBillable` – Check if a product is billable.

Synopses

```
virtual type GetValue () const =0;

virtual Bool IsAvailable () const =0;

virtual Bool IsBillable () const =0;
```

Description

This group of methods accesses data values in the WSJ database related to product information such as the product's description, name, duration, and creation.

Parameters

`Value`

The data item being requested

Table 37: Accessor Methods for VWSJProduct Class

Return Type	Method	Description and Notes
`TDate`	`GetCreateTime`	When product was created
`GUIDString`	`GetCreatorID`	GUID of product's creator
`TMoney`	`GetDurationPrice`	The cost for this product per duration unit
`EProductDuration`	`GetDurationUnits`	Bill monthly or yearly (see Table x34x)
`long`	`GetNumberResources`	Number of items in this product
`TString`	`GetProductDescription`	Free-form description of product
`GUIDString`	`GetProductID`	GUID for this product
`TString`	`GetProductName`	Name of this product

Table 37: Accessor Methods for VWSJProduct Class

Return Type	Method	Description and Notes
`HRESULT`	`GetResources (TGUIDStringVector& resourceIDs) = 0;`	List of resource GUIDs for this product
`TDate`	`GetStatusModifiedTime`	When product status was last modified
`long`	`GetSubscriptionDuration`	The default duration of this subscription

Example

The following example adds a resource to a product and uses GetProductID to obtain the product's ID in the database.

```
TGUIDStringVector resourceIDs;
resourceIDs.Append(domainPtr->GetID());
productPtr->AddResources(resourceIDs);

// Check to see if the product is valid before adding
if(FAILED(productPtr->IsValid())){
    cout << "main: Error: completed product is not valid" << endl;
    return FALSE;
}

// Commit product changes to database
if(FAILED(pProductService->UpdateProduct(context, productPtr.get()))){
    cout << "main: Error: could not update product in database" << endl;
    return FALSE;
}

productID = productPtr->GetProductID();
```

VWSJProduct::Remove*Resource*

Name

`RemoveResource, RemoveResources` – Remove resource(s) from a product.

Synopsis

```
virtual HRESULT RemoveResource (
        const GUIDString& resourceID) =0;

virtual HRESULT RemoveResources (
        const TGUIDStringVector& resourceIDs) =0;
```

Description

The `Remove` methods remove one or more resources from a product.

Parameters

`resourceID, resourceIDs`
: Input parameter. A single resource ID, or a vector of resource IDs.

Return Values

`HRESULT`

Related Methods

`VWSJProduct::AddResource`

`VWSJProduct::AddResources`

VWSJProduct::Set*Data*

Name

`Set`*`Data`* – Specify the value of a product data item.

Synopsis

```
virtual void Set Value (type Value) =0;
```

Description

This group of methods sets data values into the WSJ database related to a product.

Parameters

`Value`
The data item being set.

Table 38: Mutator Methods for VWSJProduct Class

Method	Data Parameter	Description
`SetAsAvailable`		Product is available
`SetAsBillable`		Product is billable
`SetAsNonBillable`		Product is not billable
`SetAsUnavailable`		Product is not available
`SetCreateTime`	`const TDate& time`	When product was created
`SetCreatorID`	`const GUIDString& creatorid`	User ID of person who created product
`SetDurationPrice`	`const TMoney& price`	Price of product
`SetDurationUnits`	`const EProductDuration& units`	Bill monthly or yearly
`SetProduct\ Description`	`const TString& productDescription`	Free-form description of product
`SetProductName`	`const TString& productName`	Name of product
`SetResources`	`const TGUIDStringVector& resourceID`	GUIDs of resources

Table 38: Mutator Methods for VWSJProduct Class

Method	Data Parameter	Description
`SetStatusModified\ Time`	`const TDate& time`	When product was modified
`SetSubscription\ Duration`	`const long& duration`	How long a subscription lasts

Return Values

None.

Examples

```
VWSJProduct *product = factory.MakeVWSJProduct ();
if (product == NULL) {
    return E_FAIL;
}

product->SetCreatorID(smartData.principalID);
```

The following example creates a resource and a product:

```
TSmartPtr<VWSJProduct> productPtr = factory.MakeVWSJProduct();
{
if(FAILED(productPtr->IsValid())){
    cout << "main: Error: product is not valid" << endl;
    return FALSE;
}
productPtr->SetProductName(gProductName);
productPtr->SetProductDescription("For Testing");
productPtr->SetAsAvailable();
productPtr->SetAsBillable();
productPtr->SetSubscriptionDuration(12);
productPtr->SetDurationUnits(VWSJProduct::kMonthly);
productPtr->SetDurationPrice(TMoney("USD", 59.99));
productPtr->SetCreateTime(TDate ());
productPtr->SetCreatorID(creatorID); // $$$

// Check to see if the product is valid before adding
if(FAILED(productPtr->IsValid())){
    cout << "main: Error: completed product is not valid" << endl;
    return FALSE;
}
```

VWSJProductService Class

Name

`VWSJProductService` – The service class for the WSJ product class.

Description

Methods in this class create products in the database and look them all up, or look up individual products by ID or name.

Public Methods

```
CreateProduct
GetAllProducts
LookUpBy<i>Data</i>

static Bool Test();
```

Related Classes

`VWSJProduct`

VWSJProductService::CreateProduct

Name

`CreateProduct` – Create a product.

Synopsis

```
virtual HRESULT CreateProduct(
                  const VWSJContext* ctxPtr,
                  VWSJProduct* Product) = 0;
```

Description

This method creates a product in the Transact database.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext* ctxPtr`. This is used to specify the object containing the operational context required to perform database operations.

`Product`
: Input and output. A pointer to the VWSJProduct object to be created.

Return Values

The `CreateProduct` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJProductService::GetAllProducts

Name

`GetAllProducts` – Get all products.

Synopsis

```
virtual HRESULT GetAllProducts (
                  const VWSJContext* ctxPtr,
                  TGUIDStringVector& ProductIDs) = 0;
```

Description

Get all Products.

Parameters

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Lookup methods for the VWSJProductService class. It specifies the object containing the operational context required to perform database operations.

`ProductIDs`
: Output . A vector of Product IDs returned by the method.

Return Values

The `GetAllProducts` method returns:

`S_OK`
: If the method was successful.

`E_FAIL`
: If the method was not successful.

`<DB Errors>`
: If the method resulted in database errors.

VWSJProductService::LookUpBy*Data*

Name

`LookUpBy`*`Data`* – Access a product according to value of the specified data.

Synopsis

```
virtual HRESULT LookUpBy DomainProperty ( . . . ) =0;
```

Description

This group of methods accesses domain products in the WSJ database.

Parameters

Several, and varied. See Table 39, "LookUp Accessor Methods for VWSJProductService Class," for their usage.

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. This is used by all Lookup methods for the VWSJProductService class. It specifies the object containing the operational context required to perform database operations.

`id`
: Input. A pointer to a GUIDString containing the ID of the product for which a pointer is to be returned.

`productName`
: Input. A TString containing the name of the product for which a pointer is to be returned.

`product`
: Output. A pointer to a VWSJProduct object that will contain the pointer to the product found by the LookUp methods.

Table 39: LookUp Accessor Methods for VWSJProductService Class

Method	Parameters	Description and Notes
`LookUpByID`	`const VWSJContext* ctxPtr, const GUIDString& id, VWSJProduct* product`	Return a product based on a product GUID
`LookUpByName`	`const VWSJContext* ctxPtr, const TString& productName, VWSJproduct* product`	Return a product based on its name
`LookUpBySet\ PeriodicalID`	`const VWSJContext* ctxPtr, const GUIDString& id, VWSJproduct* product`	Return a product based on its GUID

Status Returns

All the LookUp methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

Example

```
TString selectedID = formData.GetValue (kCurProducts);

VWSJProduct *productSelected = factory.MakeVWSJProduct ();
if (productSelected == NULL) {
    return E_FAIL;
}
hr = prodService->LookUpByID (wsjCtx,
                    selectedID,
                    productSelected);
if (FAILED(hr)) {
    return E_FAIL;
}
```

VWSJSubscription Class

Name

`VWSJSubscription` – The WSJ subscription class.

Description

Methods in this class center on starting, ending, and getting a subscription, along with accessing and setting product, offer, and subscriber IDs.

Enumerated Data

```
enum EWSJSubType {
    kNoSub,
   kSingle,
   kComp,
   kFree
 };
```

Table 40: Meanings of EWSJSubTypes

Value of EWSJSubType	Subscription Types
`kComp`	Complementary
`kFree`	Free
`kNoSub`	No subscription
`kSingle`	Individual buyer

Public Methods

`Get`*`Data`*
`Set`*`Data`*

```
virtual Bool IsValid() const = 0;
```

Related Classes

`VWSJSubscriptionService`

VWSJSubscription::Get*Data*

Name

Get*Data* – Access values related to a *Wall Street Journal* subscription.

Synopses

```
virtual ReturnType GetValue () const =0;

virtual ReturnType GetValue (Date) =0;
```

Description

This group of methods accesses data values in the WSJ database related to a WSJ subscription

Parameters

Value

Input value. The data item being requested.

Date

Input value. The subscription-related date requested (start, end, when paid).

Table 41: Accessor Methods for VWSJSubscription Class

Return Type	Method	Description and Notes
TMoney	GetAmountCharged	Cost of subscription charged *
TDate	GetDateCharged	When charged *
HRESULT	GetIFPEnd (TDate& ifpEndDate)	End date of Initial Free Period
GUIDString	GetOffer	GUID of this product's offer *
GUIDString	GetProduct	GUID of this product *
GUIDString	GetRequestedID	The GUID of the buyer to be operated on
HRESULT	GetStart (TDate& start)	Start date for subscription
GUIDString	GetSubscriber	Many IDs

Table 41: Accessor Methods for VWSJSubscription Class

Return Type	Method	Description and Notes
HRESULT	GetSubscriptionEnd (TDate& end)	Date subscription ends

* Specific to single type subscriptions (others are common to both single and group types)

Example

```
// Make an empty subscription piece of paper

TSmartPtr<VWSJSubscription> vsub = factory.MakeVWSJSubscription();
if (vsub.get() == NULL) {
    cout << "main:  VWSJSubscription == NULL ";
    return 1;
}

GUIDString productID, offerID; // Get offer and product guids
{

// Look up the Offer

TSmartPtr<VWSJOffer> offerPtr = factory.MakeVWSJOffer();
if(FAILED(offerPtr->IsValid())) {
    cout << "main: Error: offer is invalid" << endl;
    return FALSE;
}

TSmartPtr<VWSJOfferService> pOfferService = factory.MakeVWSJOfferService();
if(FAILED(pOfferService->LookUpByName(context, gOfferName, offerPtr.get())))
    cout << "main: Error: could not find offer name="<< gOfferName << endl;
    return FALSE;
}
offerID = offerPtr->GetOfferID();
```

VWSJSubscription::Set*Data*

Name

`Set`*`Data`* – Specify data item for a subscription.

Synopsis

```
virtual void Set Value (type Value) =0;
```

Description

This group of methods sets data values for offer, product, and subscriber ID into the WSJ database related to a subscription.

Parameters

`Value`

The data item being set.

Table 42: Mutator Methods for VWSJSubscription Class

Method	Data Parameter	Description
`SetOffer`	`const GUIDString& offer`	The GUID of the offer *
`SetProduct`	`const GUIDString& product`	The GUID of the product *
`SetRequestedID`	`const GUIDString& id`	The GUID of the subscriber to be operated on
`SetSubscriber`	`const GUIDString& subscriber`	Add a subscriber GUID

* Specific to single type subscriptions (others apply to both single and group types)

Return Values

None.

Example

```
TSmartPtr<VWSJSubscription> vsub = factory.MakeVWSJSubscription();
. . .
    offerID = offerPtr->GetOfferID();
. . .
    productID = productPtr->GetProductID();
     vsub->SetProduct(productID);
     vsub->SetOffer(offerID);// see PaidBuyerTest.cpp demo program
```

VWSJSubscriptionService Class

Name

`VWSJSubscriptionService` – Service class for WSJ subscription objects.

Description

Methods in this class look up subscription information in the database according to buyer or group ID; suspend and reinstate subscribers; cancel groups and individuals; specify how much to charge for a subscription; and specify that a subscriber can not renew.

Public Methods

`LookUp`*`Data`*
`SetDoNotRenew`

Related Classes

`VWSJSubscriber`

VWSJSubscriptionService::LookUp*Data*

Name

`LookUp`*`Data`* – Look up WSJ subscriptions.

Synopsis

```
virtual HRESULT LookUpSubscriptionDataByID ( . . . ) =0;
```

Description

This group of methods looks up and returns a list of one or more subscriptions.

Parameters

Several, and varied. See Table 43, "LookUp Accessor Methods for VWSJSubscriptionService Class," for their usage.

`ctxPtr`
: Input. A pointer to a VWSJContext constant, i.e., `const VWSJContext * ctxPtr`. It specifies the object containing the operational context required to perform database operations.

`wsjSubscription`
: Input. A pointer to a VWSJSubscription object. constant, i.e., `const VWSJBuyer* buyer`. This is input to all LookUp methods for the VWSJSubscriptionService class.

`status`
: Input. The status for group subscription information.

`subscription`
: Output. The returned subscription information.

`subscriptionIDVector`
: Output. A list of TGroupSubscription IDs.

Table 43: LookUp Accessor Methods for VWSJSubscriptionService Class

Method	Parameters	Description and Notes

`LookUpByID`	`const VWSJContext* ctxPtr,`
	`const GUIDString& id`
	`const VWSJSubscription`
	`::EWSJSubType& subType,`
	`TVWSJSubscription*`
	`vwsjSubscription`
		Return WSJ subscription object based on GUID

`LookUpGroupByBuyerID`	`const VWSJContext* ctxPtr,`
	`const TGroupSubscription`
	`::EGrpSubStatus status,`
	`const VWSJSubscription*`
	`wsjSubscription,`
	`TGUIDStringVector&`
	`groupSubscriptionIDVector`
		Return list of group subscription GUIDs based on buyer GUID

`LookUpIndividual\`
`ByBuyerID`
	`const VWSJContext* ctxPtr,`
	`const VWSJSubscription*`
	`wsjSubscription,`
	`TGUIDStringVector&`
	`subscriptionIDVector`
		Return list of subscription GUIDs based on buyer GUID

Status Returns

All the LookUp methods return the same values:

`S_OK`
: If the method succeeded.

`E_FAIL`
: If the method failed.

`<DB Errors>`
: Database-related error status codes.

APPENDIX A

WSJGetUsernamePassword

Name

WSJGetUsernamePassword – Provide username and password to Transact.

Synopsis

```
int WSJGetUsernamePassword (
                    WSJHTTPRequest *reqPtr,
                    char **username,
                    char **plaintext_password);
```

Description

This function provides a valid username and plaintext password to Transact from a location or means different from the standard Transact/WSJIE cookies and tickets.

If the username/password combination is not valid, an auth failed error screen will be generated with enough context to be able to discern (with server side JavaScript) whether the failure was from the WSJIE function or from Transact.

Enumerated Data

```
Enum WSJLoginStatus {
          WSJPassThrough
          WSJOverRide };
```

Table 44: Meanings of Status Types

Value of WSJLoginStatus	Meaning of Status
`WSJPassThrough`	Username and password could not be found.
`WSJOverRide`	Username and password were found.

Parameters

`reqPtr`
: Input.

`username`
: Output. A pointer to the username.

`plaintext_password`
: Output. A pointer to the user's password.

Return Values

The `WSJGetUsernamePassword` function returns:

`WSJOverride`
: If the username and password were returned.

`WSJPassThrough`
: If the username and password could not be returned.

Notes

It is the caller's responsibility to call delete( ) on the returned username and password.

For auto login, Transact will store enough information in a cookie to automatically identify and authenticate the user. This cookie's name and duration will be determined through registry settings:

- WSJIE\AUTOLOGIN\ CookieName
- (WSJIE\AUTOLOGIN\ CookieDuration)

The header files required to use this function are as follows:

- WSJHTTPRequest.h -- abstracted from Transact's httprequest
- WSJHTTPCookie.h - abstracted from Transact's httpcookie
- WSJHTTPSetCookie.h - abstracted from Transact's httpsetcookie

Glossary

accessor methods
: Methods that access, look up, or get items from the Transact database.

applications
: Custom Buyer and Admin applications.

buyer
: A person (or other operator of a Web client) who can make a purchase (or subscribe to an offer available to a group) using a browser.

concrete classes
: Classes whose names begin with "T," for example, TWSJAccessHistory.

CSR
: Customer Service Representative (CSR), analogous to Transact's Employee role, but with more functionality and fewer permissions.

external systems
: Legacy components such as the WSJ NR and OLF billing systems.

framework classes
: Low-level, custom classes to provide additional, custom Transact functionality, including the API to which Dow Jones employees will program Seller applications. All higher-level applications should write to this framework, as certain function calls will cause interactions with eternal components tht are not supported by the analogous core Transact calls.

group
: One or more users who share the same group ID. A user may be a member of more than one group. Group members may be other groups.

GUID
: Globally Unique Identifier. A 128-bit (16 bytes) integer guaranteed to be unique. In the COM world, clients find servers on the basis of the server's GUID. For more details, refer to a COM tutorial. Call `GenerateGUID` to create a GUID.

GUID string
: A data type used by the WSJIE Seller API to represent GUIDs. A GUID string is a `TString` containing a 38-character representation of a GUID. For details on GUID strings, see the *Transact 4 Utility Classes Programmers Guide*.

module
: The term used to describe applications, plugins, libraries, and generally any piece of development work that has been packaged off to a particular developer or development team.

mutator methods
: Methods that change a value in the Transact database, typically setting it specifically via a `Set`*`Data`* method.

NR
: Another WSJ proprietary billing system.

OLF
: WSJ's proprietary billing system to which this API integrates.

plugins
: Addditional required functionality not provided by WSJ framework classes.

SDK
: Software Developer's Kit (SDK). A programming package that typically contains APIs, tools, and documentation, enabling a programmer to develop software applications for a particular platform. Sometimes referred to as a "software development kit."

state classes
: Custom classes to communicate to external systems.

Transact database
: A database in which all persistent information is stored. Each payment instruction object is essentially one row of the Transact database. As an application programmer, you never access the Transact database directly. Rather, you access it indirectly through accessor mutator methods.

virtual classes
: Classes whose names begin with "V," for example, VWSJBuyer.

WSJIE
: Wall Street Journal Interactive Edition (WSJIE).

Index

G

H

I

L

M

N

O

P

R

S

T

U

V

W