StockTrader Business Service

Description

The StockTrader Business Service is responsible for providing an interface to clients for the state of the application as a whole. It provides access to data within the StockTrader Database directly, as well as passing off buy and sell orders to the queue managed by the StockTrader Order Processor Service. Nearly all interactions (including read-only) interactions with any data managed by the Stonehenge StockTrader Sample Application, will flow through the Business Service at some point. This excludes only overall application configuration which is managed by a separate service.

Table of Contents

Live Demonstrations

In December of 2009, the PHP and .NET implementations of the StockTrader Business Service were ported to run on Windows Azure. These live cloud-based versions of the StockTrader Business Service can be accessed using the URLs in the table below:

Implementation	URL
PHP	http://wso2wsastest1.cloudapp.net/php_stocktrader/business_service/business_svc.php
.NET	http://stocktraderazure.cloudapp.net:8080/tradebusinessservice.svc

Service Operations

The port **ITradeServices** which exposes these operations is located under the namespace http://trade.samples.websphere.ibm.com. This is due to historical reasons, and was originally done to ensure compatibility between the .NET Implementation of the StockTrader sample and IBM's original implementation of the same. This was before the StockTrader sample became a part of Apache Stonehenge.

isOnline

The **isOnline** operation is a one-way service call that is used by clients of this service to determine if the service is running. It is employed in StockTrader load-balanced scenarios to ensure application-level failover of service-to-service remote calls to clusters running this service.

Parameter	Type	Description	Direction
loginReturn	Void	N/A	out

login (deprecated in M2)

The **login** operation is used to indicate that a login has occurred. It returns information about the account associated with the userID passed in. In the case of an error, the error message can be checked to determine why the login failed.

In those implementations supporting claims based security, this service operation is not explicitly called whenever a user logs in. In subsequent service calls that have a userID parameter, an empty string is sent instead, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Identifier of the user that is logging in	in
password	String	Password of the user that is logging in	in
loginReturn	AccountDataBean	Account data for the user that is logging in	out

getOrders

The **getOrders** operation is used to retrieve recent orders placed by a given users. Orders are considered to be user instructions to buy **or** sell a stock. This means that not every order will have a negative impact on the user's account balance. Additionally sell orders are not associated with a specific holding, as they may include only a portion of an existing holding. Buy orders, on the other hand, will always become a new holding.

Each implementations can control the number of orders returned, and it is not guaranteed to be a complete list of all orders ever placed. Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Identifier of the user for which to retrieve orders	in
getOrdersReturn	Array of OrderDataBean	List of the orders placed by the user	out

getAccountData

The **getAccountData** operation is used to retrieve information about an account associated with a given userID. This data will contain purely statistical information about the account itself. In order to get information about the *user* specifically, the **getAccountProfileData** operation should be used instead.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Identifier of the user for which to get account data	in
getAccountDataReturn	AccountDataBean	Account data for the user indiciated	out

getAccountProfileData

The **getAccountProfileData** operation is used to retrieve authentication, contact, and payment information for a given user, based on the user's userID.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Identifier of the user for which to get account profile data	in
getAccountProfileDataReturn	AccountProfileDataBean	Account profile data for the user indiciated	out

updateAccountProfile

The **updateAccountProfile** operation is used to update account profile data for a given user with the data supplied. All data can be changed for a given user, except for the user's userID.

Parameter	Type	Description	Direction
profileData	AccountProfileDataBean	Updated account profile data to apply to an account	in
updateAccountProfileReturn	AccountProfileDataBean	Updated account profile data that was passed in	out

logout (deprecated in M2)

The **logout** operation is used to signal that a user has logged out of the client application.

In those implementations supporting claims based security, this service operation is not explicitly called whenever a user logs out.

Parameter	Type	Description	Direction

userID	String	Unique identifier of the user to log out of the system	in
logoutReturn	Void	N/A	out

buy

The **buy** operation is used to asynchronously register a buy order with the StockTrader Order Processor Service. By submitting an order through this operation, there is no guarantees that the order will be successfully completed.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user that is placing an order	in
symbol	String	Symbol for the stock of which the user is buying shares	in
quantity	Floating Point Number	Number of shares of the stock that the user is buying	in
orderProcessingMode	Integer	This parameter is no longer used, but instead is provided through configuration	in
buyReturn	OrderDataBean	Order generated as a result of the request	out

sell (deprecated)

The **sell** operation is used to asynchronously register a sell order with the StockTrader Order Processor Service. By submitting an order through this operation, there is no guarantees that the order will be successfully completed.

This operation differs from the **sellEnhanced** operation in that this operation always sells the entire holding. This operation has been retained for backwards compatibility with IBM Trade 6.1, as all implementations of the Stonehenge StockTrader Sample Application that are a part of the Apache Stonehenge project use the **sellEnhanced** operation.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user that is placing an order	in
holdingID	Integer	Unique identifier of the holding that the user is selling	in
orderProcessingMode	Integer	This parameter is no longer used, but instead is provided through configuration	in
sellReturn	OrderDataBean	Order generated as a result of the request	out

getHoldings

The **getHoldings** operation is used to retrieve a list of the user's holdings.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user for which to get holdings	in
getHoldingsReturn	Array of HoldingDataBean	A list of holdings associated with the user	out

register

The **register** operation is used to create an account and a new user within the StockTrader application.

Those implementations that support claims-based security will not rely on the password used here for user authentication. A Passive STS will be used for authentication of a user with a third party. The claims passed from that Passive STS will determine whether or not the user is authorized to perform the actions requested.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user for which to register an account	in
password	String	Password that the user must use for authentication	in
fullname	String	Full name of the user	in
address	String	Billing address of the user	in
email	String	Email address of the user	in
creditcard	String	Credit card number of the user	in

openBalance	Floating Point Number	Opening balance for the account associated with the user	in
registerReturn	AccountDataBean	Account created as a result of the service call	out

getClosedOrders

The **getClosedOrders** operation is used to get a list of the orders where the status is set to **closed**. This allows orders to be displayed to the user when accessing the StockTrader Client. Orders returned as a result of calling this operation will be marked as **completed**, and will not be reported again by subsequent calls.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user for which to retrieve closed orders	in
getClosedOrdersReturn	Array of OrderDataBean	List of orders that have closed	out

getMarketSummary

The **getMarketSummary** operation is used to retrieve a summary of the current market conditions.

Parameter	Type	Description	Direction
getMarketSummaryReturn	MarketSummaryDataBeanWS	Contains the current market summary	out

getQuote

The **getQuote** operation is used to generate a quote for a given stock.

Parameter	Type	Description	Direction
symbol	String	The symbol of the stock for which to generate a quote	in
getQuoteReturn	QuoteDataBean	Quote for the stock requested	out

getHolding

The **getHolding** operation is used to retrieve details about a specific holding of a given user. Even though the holdingID should be unique for all holdings, a userID is supplied for this service call as well to ensure absolutely that the correct holding is located.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user that is associated with the account that the holding has been associated with as well	in
holdingID	Integer	Unique identifier of the holding for which to retrieve information	in
getHoldingReturn	HoldingDataBean	Holding that was requested	out

getTopOrders

The **getTopOrders** operation is used to retrieve the top *N* orders for a given user. Each implementation is responsible for determining the number of orders that will be returned.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user for which the top orders should be retrieved	in
getTopOrdersReturn	Array of OrderDataBean	List of the top orders made by the given user	out

sellEnhanced

The **sellEnhanced** operation is used to asynchronously register a sell order with the StockTrader Order Processor Service. By submitting an order through this operation, there is no guarantees that the order will be successfully completed.

This operation differs from the **sell** operation in that this operation allows for the partial sale of holdings. All implementations of the Stonehenge StockTrader Sample Application that are a part of the Apache Stonehenge project use the **sellEnhanced** operation as opposed to the **sell** operation for handling the sale of holdings.

Those implementations that support claims-based security will accept an empty string in place of a userID, since the userID can be deduced from the claims included as part of the service call.

Parameter	Type	Description	Direction
userID	String	Unique identifier of the user that is placing an order	in
holdingID	Integer	Unique identifier of the holding that the user is selling	in
quantity	Floating Point Number	Number of shares of the stock that the user is selling	in
sellEnhancedReturn	OrderDataBean	Order generated as a result of the request	out