Chicago Stock Exchange

CHX Risk Management Tools
Web Interface Protocol

Version 1.0

November 8, 2013

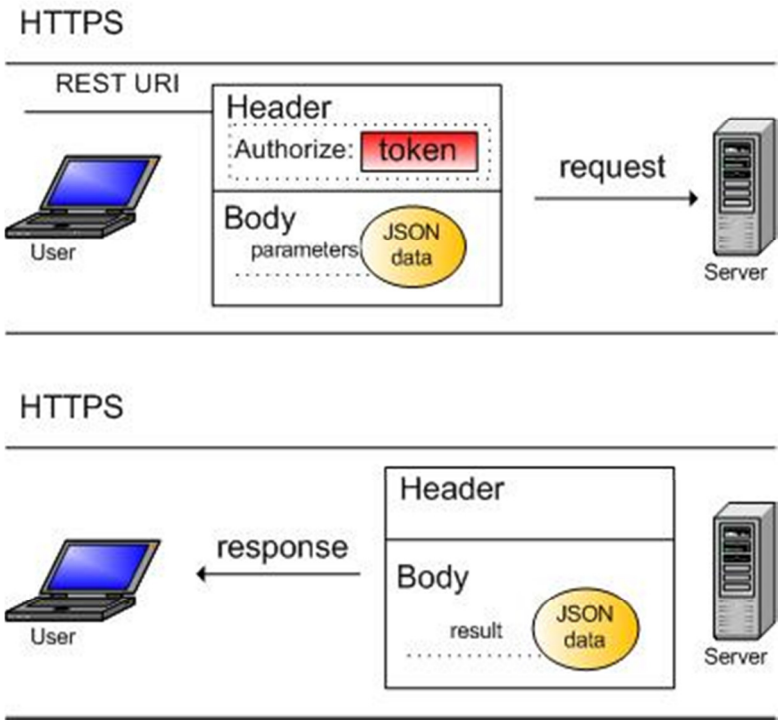
Contents

Overview

Currently the CHX Web Interface Protocol only supports Kill Switch functionality. The CHX Web Interface Protocol is based on the stateless REST interface to Kill Switch objects over a secured HTTPS protocol. Each request (except for “login”) is authorized by a security token obtained from the “login” request. The security token is stored in the HTTP header of each secured request as a value of standard “Authorization” HTTP header property.

The security token is a character string generated by the Kill Switch Server and once generated, remains the same for the user until the “logout” request is issued. Each request parameter is passed inside the HTTP request body in the form of a JSON object. Parameters are passed only for POST and PUT HTTP methods.

Concurrency control is implemented using “ETag” and “If-Match” HTTP header properties. Each PUT/DELETE request contains an “If-Match” property populated with the last update time for a Clearing Firm/Trading Firm. Each response returns an “ETag” property with the last update time (in ms since 01/01/1970) for the entity stored in the web service. If the Web Service finds that the lastUpdateTime in the PUT/DELETE request is before the last update time stored on Web Service, then the request is declined with HTTP status 412 (pre-condition failed) – it means that another user updated the same entity and an update can’t be performed without a preliminary data refresh.



Each response is passed as a JSON object in the response body and contains 4 properties: “code” “msg”, “data” and “lastUpdateTime”. The “code” property contains a message status code, similar to HTTP status codes. The “msg” property contains error message text related to a given error code. The “data” property contains Kill Switch object data in JSON format. The “lastUpdateTime” property contains the last update time (in milliseconds since 01/01/1970) and is used for concurrency control. This representation allows for the passing of information about a possible error when the data object is not available.

Data Model

There are several main collections of data objects used in the Kill Switch Server environment: Clearing Firms, Trading Firms, Distribution Lists and User Information. The object of each collection consists of a set of properties which are either strings values or other object values.

Clearing Firm Object

- **id** (string) – ID of a given Clearing Firm.
- **name** (string) – the Clearing Firm Name.
- **tradingFirms** (collection) – the set of Trading Firm objects related to given Clearing Firm.
- **lists** (collection) – the set of Distribution List objects related to given Clearing Firm.

Trading Firm Object

- **id** (string) – ID of a given Trading Firm.
- **name** (string) – the Trading Firm Name.
- **profile** (Profile object) – the latest values of Trading Firm profile parameters.
- **state** (string) – current state of the Trading Firm.
- **currentNv** (string) – the current calculated Notional value for the Trading Firm.
- **cfMaxValues** (collection) – set of Max Notional Value’s set by Clearing Firms for the given Trading Firm.
- **cfNames** (collection) – set of Clearing Firms Names in the same order as cfMaxValues.
- **lists** (collection) – the set of Distribution List objects related to the given Trading Firm.

Distribution List Object

- **id** (string) – ID of the given Distribution List object
- **name** (string) – the Distribution List Name.
- **content** (object) – the Distribution List Content object reflecting the content of the given Distribution List.

User Object

- **fullName** (string) – the user's full name.
- **allowAutoActions** (boolean) – true if the automatic actions column should be visible for the given user's account.
- **role** (string) – the user's role. Can be a "clearing_firm" or "trading_firm".
- **token** (string) – the latest valid security token for the given user.
- **email** (string) – the email address associated with the user's account.
- **temporaryPassword** (boolean) – TRUE if a temporary password is set, otherwise FALSE.
- **locked** (boolean) – TRUE if the user's account is locked, otherwise FALSE.
- **error** (string) – the error message explained why user can't be provided

Other objects participated in the Data Model as part of main objects or methods parameters:

Credentials Object (used for login to the Web service)

- **login** (string) – user's login name.
- **password** (string) – user's password.

Distribution List Content Object

- **emails** (collection) – the set of email addresses associated with the given Distribution List.

Profile Object

- **maxNv** (string) – the Maximum Notional Value (in $).
- **autoAction** (integer) – action which will be automatically performed: 0 – Notify Only, 1 – Shutoff Firm, 2 – Cancel Orders, 3 – Shutoff Firm and Cancel Orders.
- **expLimit1** (string) – Exposure limit 1 (in %).
- **distList1** (string) – ID of Distribution List associated with expLimit1.

- **expLimit2** (string) – Exposure limit 2 (in %).
- **distList2** (string) – ID of Distribution List associated with expLimit2.
- **expLimit3** (string) – Exposure limit 3 (in %).
- **distList3** (string) – ID of Distribution List associated with expLimit3.

Operations

General data manipulation operations are exposed by the REST interface using standard HTTP methods:

- HTTP.GET – to retrieve data object by its ID or collection of objects.
- HTTP.POST – create new element or insert new element into collection.
- HTTP.PUT – update properties of existing object.
- HTTP.DELETE – delete object or remove object from collection by its ID.

Object or collections are addressed by a URL containing the path to the object and it's ID (including id(s) of parent object(s)). POST and PUT methods also contains JSON data inside the request body which are used as the parameter of the corresponding insert/update operation.

There are 10 custom operations within the Kill Switch Web Protocol:

- **login** – authorization to Web Service and obtain security token for user.
- **logout** – end communication session and invalidate the last used security token.
- **sendLoginByEmail** – sends login name to user's email address associated with their account.
- **sendTemporaryPassword** – sends a new temporary password to user's email address associated with their account.
- **changePassword** – changes the user's password to a newly specified password.
- **shutoff** – Shutoff Trading Firm (stop accepting new orders).
- **resume** – Resume Trading Firm (if Trading Firm is in a shutoff state).
- **cancel** – Cancel all open orders for the specified Trading Firm.
- **shutoff and cancel** – Shutoff the Trading Firm and cancel all its open orders.
- **reset** – Reset the Calculated Notional Value for the specified Trading Firm to zero.

They are implemented as POST and GET operations of corresponding objects to which they are applied.

Implementation

Login and Logout

Operation: login – returns user object with a security token if authorized; otherwise the user object contains an error message with a reason.

URI: /api/users/login
HTTP Method: POST
Parameter: Credentials (in request body)

Example:

Request:

uri: /api/users/login
request body: { “login”: “user1”, “password” : “changeit” }

Response:

response body: { “fullName” : “John Smith”, “allowAuto” : true,
“role” : “clearing_firm”, “token” : “jsdjs232k”, “error” : “” }

Operation: logout – invalidates the user’s security token and returns TRUE if successful, otherwise it returns FALSE.

URI: /api/users/logout
HTTP Method: POST
Parameter: none

Example:

Request:

uri: /api/users/logout

header property: “Authorization”: “token1”

Response:

response body: true

Operation: send login name by email – sends an email to the user with their login name. User is found by email address associated with their account. If the user is found and an email was successfully sent, then the method returns “true”, otherwise an error message is returned. This request should not be authorized by a token.

URI: /api/users/emaillogin
HTTP Method: POST
Parameter: request body – email address associated with user's account

Example:

Request:

uri: /api/users/emaillogin
request body: "test@example.com"

Response:

response body: "true"

Operation: get security questions – returns the list of security questions available for a given user name or null if the user login name does not exist. If the passed user login name is null then all security questions are returned. If the passed user login name is not null, then only those questions with user provided non-empty answers will be returned. This request should not be authorized by token.

URI: /api/users/questions
HTTP Method: GET
Parameter: header property "Authorization" – user's login name

Example:

Request:

uri: /api/users/questions
header property: "Authorization": "user1"

Response:

response body: ["Security question 1", "Security question 2", "Security question 3"]

Operation: get user's answer on security question – returns the user's answer to security question by its ID. An empty answer is returned for an unanswered question. It returns "true" on success, otherwise it returns an error message. User's security token must be provided.

URI: /api/users/answers/{securityQuestionId}
HTTP Method: GET
Parameter: securityQuestionId – ID (order number) of security question

Example:

Request:

uri: /api/users/answers/2

header property: “Authorization”: “token1”

Response:

response body: “some answer”

Operation: set answer on security question – sets answer on specified security question (empty string sets question to unanswered). At least one security question should have an answer after update. It returns “true” on success, otherwise an error message is returned. User’s security token must be provided.

URI: /api/users/answers/{securityQuestionId}
HTTP Method: PUT
Parameters:
securityQuestionId – ID (order number) of security question
request body – answer on given security question

Example:

Request:

uri: /api/users/answers/2
header property: “Authorization”: “token1”
request body: “new answer”

Response:

response body: “true”

Operation: send temporary password to user – if user forgot their password and correctly answered a security question, a new temporary password will be created and stored on the users account. Then an email with their new temporary password will be sent to the email address associated with their account. User is found by their login name associated with his account. If the user is found and an email was successfully sent, then the method returns “true”, otherwise an error message is returned. This request should not be authorized by token.

URI: /api/users/questions/{securityQuestionId}/sendtemp
HTTP Method: POST
Parameters:
securityQuestionId – ID of security question to answer on
request body – user’s answer on security question

Example:

Request:

uri: /api/users/questions/1/sendtemp
request body: “answer on question 1”

Response:

response body: “true”

Operation: change user’s password – changes the user’s password. It returns “true” on success, otherwise an error message is returned. The request should not be authorized by token.

URI: /api/users/password
HTTP Method: PUT
Parameter: request body – user’s old password and new password

Example:

Request:

uri: /api/users/password
request body: { loginName: oldPassword, password: newPassword }

Response:

response body: “true”

Operation: update email – updates the user’s email to newly provided value. It returns “true” on success, otherwise an error message is returned.

URI: /api/users/email
HTTP Method: PUT
Parameter: request body – new email address to be set

Example:

Request:

uri: /api/users/email
header property: “Authorization”: “token1”
request body: “new@example.com”

Response:

response body: “true”

Clearing Firm Operations

Clearing Firm operations are intended for use by Clearing Firm user roles only. Trading Firm user roles have no access to Clearing Firm operations.

Operation: Get Clearing Firms list – returns a list with only their Clearing Firm's data.

URI: /api/clearing
HTTP Method: GET
Parameter: none

Example:

Request:

uri: /api/clearing
header property: "Authorization": "token1"

Response:

response body: [{ "id" : "2", "name" : "Clearing Firm 2", …}]
header parameter "ETag": last update time of Trading Firms collection

Operation: get Clearing Firm Info – returns full information about the Clearing Firm having specified an ID (including data profiles and distribution lists for nested Trading Firms).

URI: /api/clearing/{clearingFirmId}
HTTP Method: GET
Parameter: clearingFirmId – ID of Clearing Firm to get information about

Example:

Request:

uri: /api/clearing/1
header property: "Authorization": "token1"

Response:

response body: { "id" : "1", "name" : "Clearing Firm 1", …}
header parameter "ETag": last update time of Trading Firms collection

Operation: update Clearing Firm profile – updates parameters for a Clearing Firm data profile for specified Trading Firm. It returns "true" if the update was successful, otherwise an error message is returned.

URI: /api/clearing/{clearingFirmId}/trading/{tradingFirmId}/profile
HTTP Method: PUT
Parameters:
clearingFirmId – ID of enclosed Clearing Firm
tradingFirmId – ID of enclosed Trading Firm whose profile will be updated
request body – Clearing Firm profile data

Example:

Request:

uri: /api/clearing/2/trading/3/profile
header property: "Authorization": "token1"
request body: { "maxNv": "2.0", "autoAction":1, "exp1":"50",… }
header parameter "If-Match": last update time of Trading Firms collection

Response:

response body: "true"
header parameter "ETag": last update time of Trading Firms collection

Operation: get Clearing Firm Distribution Lists – returns a collection of all Distributions Lists available for the given Clearing Firm.

URI: /api/clearing/{clearingFirmId}/dlists
HTTP Method: GET
Parameter: clearingFirmId – ID of the Clearing Firm to get Distributions Lists for

Example:

Request:

uri: /api/clearing/1/dlists
header property: "Authorization": "token1"

Response:

response body: [{ "id" : "1", "name" : "Distribution List 1"},…]
header parameter "ETag": last update time of Distribution Lists collection

Operation: create Clearing Firm Distribution List – creates a new Distribution List, adds it to the set of available ones for a given Clearing Firm and returns its ID. On error, an error message started with "!" will be returned.

URI: /api/clearing/{clearingFirmId}/dlists
HTTP Method: POST
Parameters:
clearingFirmId – ID of Clearing Firm to create distribution lists for
request body – new Distribution List name

Example:
Request:
uri: /api/clearing/1/dlists
header property: "Authorization": "token1"
request body: "new_list_name"

Response:
response body: "id21"

Operation: shutoff Clearing Firm form – shutoff Trading Firm by the Clearing Firm. It returns "true" on success, otherwise an error message is returned.

URI: /api/clearing/{clearingFirmId}/trading/{tradingFirmId}/shutoff
HTTP Method: POST
Parameters:
clearingFirmId – ID of enclosing Clearing Firm
tradingFirmId – ID of the Trading Firm to shutoff by Clearing Firm

Example:
Request:
uri: /api/clearing/1/trading/2/shutoff
header property: "Authorization": "token1"

Response:
response body: "true"

Operation: resume Clearing Firm form – resume Trading Firm by Clearing Firm from a shutoff state. It returns "true" on success, otherwise an error message is returned.

URI: /api/clearing/{clearingFirmId}/trading/{tradingFirmId}/resume
HTTP Method: POST

Parameters:
clearingFirmId – ID of enclosed Clearing Firm
tradingFirmId – ID of Trading Firm to resume by Clearing Firm

Example:
Request:
uri: /api/clearing/1/trading/2/resume
header property: “Authorization”: “token1”

Response:
response body: “true”

Operation: cancel Clearing Firm orders – cancel Trading Firm orders which were set by Clearing Firm. It returns “true” on success, otherwise an error message is returned.

URI: /api/clearing/{clearingFirmId}/trading/{tradingFirmId}/cancel
HTTP Method: POST
Parameters:
clearingFirmId – ID of enclosed Clearing Firm
tradingFirmId – ID of Trading Firm to cancel orders by Clearing Firm

Example:
Request:
uri: /api/clearing/1/trading/2/cancel
header property: “Authorization”: “token1”

Response:
response body: “true”

Operation: shutoff Clearing Firm form and cancel orders – shutoff Trading Firm and cancel orders by Clearing Firm. It returns “true” on success, otherwise an error message is returned.

URI: /api/clearing/{clearingFirmId}/trading/{tradingFirmId}/shutoffcancel
HTTP Method: POST
Parameters:
clearingFirmId – ID of enclosed Clearing Firm
tradingFirmId – ID of Trading Firm to shutoff and cancel orders by Clearing Firm

Example:
Request:
uri: /api/clearing/1/trading/2/shutoffcancel
header property: “Authorization”: “token1”

Response:

response body: “true”

Operation: reset Clearing Firm Calculated Notional Value – resets the Calculated Notional Value for Trading Firm to zero which was provided by Clearing Firm. It returns “true” on success, otherwise an error message is returned.

URI: /api/clearing/{clearingFirmId}/trading/{tradingFirmId}/reset
HTTP Method: POST
Parameters:
clearingFirmId – ID of enclosed Clearing Firm
tradingFirmId – ID of Trading Firm to reset current Calculated Notional Value set by Clearing Firm

Example:

Request:

uri: /api/clearing/1/trading/2/reset
header property: “Authorization”: “token1”

Response:

response body: “true”

Trading Firm Operations

Trading Firm operations are intended for use by Trading Firm user roles only. Clearing Firm user roles have no access to Trading Firm operations.

Operation: get Trading Firms – return a list of available Trading Firm information for a Trading Firm user or Clearing Firm user.

URI: /api/trading
HTTP Method: GET
Parameter: none

Example:

Request:

uri: /api/trading
header property: “Authorization”: “token1”

Response:

response body: [{ “id” : “25”, “name” : “Trading Firm 3”,… }]
header parameter “ETag”: last update time of Trading Firm profile

Operation: get Trading Firm Info - returns all information about a Trading Firm having specified an ID.

URI: /api/trading/{tradingFirmId}
HTTP Method: GET
Parameter: tradingFirmId – ID of Trading Firm to get information about

Example:

Request:

uri: /api/trading/26
header property: “Authorization”: “token1”

Response:

response body: { “id” : “26”, “name” : “Trading Firm 4”,… }
header parameter “ETag”: last update time of Trading Firm profile

Operation: update Trading Firm profile – updates parameters for a Trading Firm data profile for the specified Trading Firm. It returns “true” if the update was successful, otherwise an error message is returned.

URI: /api/trading/{tradingFirmId}/profile
HTTP Method: PUT
Parameters:
tradingFirmId – ID of enclosed Trading Firm whose profile will be updated
request body – Trading Firm profile data

Example:

Request:

uri: /api/trading/2/profile
request body: { “maxNv”: “2.0”, “autoAction”:1, “exp1”:”50”,… }
header parameter “If-Match”: last update time of Trading Firm profile
header property: “Authorization”: “token1”

Response:

response body: true
header parameter “ETag”: last update time of Trading Firm profile

Operation: get Trading Firm Distribution Lists – returns a collection of all Distributions Lists available for given Trading Firm.

URI: /api/trading/{tradingFirmId}/dlists
HTTP Method: GET
Parameter: tradingFirmId – ID of Trading Firm to get Distributions Lists for

Example:

Request:

uri: /api/trading/1/dlists
header property: “Authorization”: “token1”

Response:

response body: [{ “id” : ”3”, “name” : “Distribution List 5”},…]
header parameter “ETag”: last update time of Distribution Lists collection

Operation: create Clearing Firm Distribution List – creates a new Distribution List, adds it to the set of available Distribution Lists for the specified Trading Firm and returns it’s ID. On error it returns an error message starting with a “!”.

URI: /api/trading/{tradingFirmId}/dlists
HTTP Method: POST
Parameters:
tradingFirmId – ID of Trading Firm to create distribution lists for
request body – new Distribution List name

Example:

Request:

uri: /api/trading/1/dlists
request body: “new_list_name”
header property: “Authorization”: “token1”

Response:

response body: “id25”

Operation: shutoff Trading Firm form – shutoff Trading Firm. It returns “true” on success, otherwise an error message is returned.

URI: /api/ trading/{tradingFirmId}/shutoff
HTTP Method: POST
Parameter: tradingFirmId – ID of Trading Firm to shutoff

Example:

Request:

uri: /api/ trading/2/shutoff
header property: “Authorization”: “token1”

Response:

response body: “ true”

Operation: resume Trading Firm form – resume Trading Firm which was previously shutoff. . It returns “true” on success, otherwise an error message is returned.

URI: /api/trading/{tradingFirmId}/resume
HTTP Method: POST
Parameter: tradingFirmId – ID of Trading Firm to resume

Example:

Request:

uri: /api/trading/2/resume
header property: “Authorization”: “token1”

Response:

response body: “true”

Operation: cancel Trading Firm orders – cancel Trading Firm orders for the specified Trading Firm. . It returns "true" on success, otherwise an error message is returned.

URI: /api/ trading/{tradingFirmId}/cancel
HTTP Method: POST
Parameter: tradingFirmId – ID of Trading Firm to cancel their orders

Example:

Request:

uri: /api/ trading/2/cancel
header property: "Authorization": "token1"

Response:

response body: "true"

Operation: shutoff Trading Firm and cancel orders – shutoff Trading Firm and Cancel Orders for the specified Trading Firm. It returns "true" on success, otherwise an error message is returned.

URI: /api/ trading/{tradingFirmId}/shutoffcancel
HTTP Method: POST
Parameter: tradingFirmId – ID of Trading Firm to shutoff and cancel orders

Example:

Request:

uri: /api/ trading/2/shutoffcancel
header property: "Authorization": "token1"

Response:

response body: "true"

Operation: reset Trading Firm Calculated Notional Value - resets the Trading Firms Calculated Notional Value to zero. . It returns "true" on success, otherwise an error message is returned.

URI: /api/trading/{tradingFirmId}/reset
HTTP Method: POST
Parameter: tradingFirmId – ID of Trading Firm to reset Calculated Notional Value

Example:

Request:

uri: /api/trading/2/reset

header property: “Authorization”: “token1”

Response:

response body: “true”

General Operations with Distribution Lists

General operations with Distribution Lists are related to modifications of their content and rename/remove Distribution List objects themselves. These operations are available to any user which has access to a given Distribution List.

Operation: get Distribution List content – returns the value of the Distribution List content.

URI: /api/dlists/{listId}/content
HTTP Method: GET
Parameter: listId – ID of Distribution List to get content from

Example:

Request:

uri: /api/dlists/2/content
header property: “Authorization”: “token1”

Response:

response body: {“emails” : [“a@b.com”, “c@d.com”]}
header parameter “ETag”: last update time of Distribution List object

Operation: update Distribution List content – updates the value of the Distribution List content and returns “true” on success, otherwise an error message is returned.

URI: /api/dlists/{listId}/content
HTTP Method: PUT
Parameters:
listId – ID of Distribution List to update content of
request body – String value with new content (space, comma or semicolon –separated list of emails)

Example:

Request:

uri: /api/dlists/2/content
request body: a@b.com, c@d.com e@f.com
header parameter “If-Match”: last update time of Distribution List object
header property: “Authorization”: “token1”

Response:

response body: “true”
header parameter “ETag”: last update time of Distribution List object

Operation: rename Distribution List – rename the Distribution List and return “true” if success, otherwise an error message is returned.

URI: /api/dlists/{listId}
HTTP Method: PUT
Parameters:
listId – ID of Distribution List to rename
request body – String value of new name

Example:

Request:

uri: /api/dlists/2
request body: “new name”
header parameter “If-Match”: last update time of Distribution Lists collection
header property: “Authorization”: “token1”

Response:

response body: “true”
header parameter “ETag”: last update time of Distribution Lists collection

Operation: remove Distribution List – delete the Distribution List and return “true” if success, otherwise an error message is returned.

URI: /api/dlists/{listId}
HTTP Method: DELETE
Parameter: listId – ID of Distribution List to remove

Example:

Request:

uri: /api/dlists/2
header parameter “If-Match”: last update time of Distribution Lists collection
header property: “Authorization”: “token1”

Response:

response body: “true”
header parameter “ETag”: last update time of Distribution Lists collection

Web Service Client Interface

The Java library is provided for use in the Web Service over REST calls. It is located in the WebCommon module and it contains data model classes and implementation of the client interface to call Web Service methods.

```
public interface WebService {

        Message<UserDTO> login(String login, String password);
        Message<Boolean> logout(String token);

        Message<String> sendLoginNameByEmail(String emailAddress);
        Message<List<String>> getSecurityQuestions(String loginName);
        Message<String> getSecurityAnswer(String token, String securityQuestionId);
        Message<String> setSecurityAnswer(String token, String securityQuestionId, String newAnswer);
        Message<String> sendTemporaryPassword(String loginName, String securityQuestionId, String securityAnswer);
        Message<String> changePassword(String token, String oldPassword, String newPassword);
        Message<String> updateEmail(String token, String email);

        Message<List<ClearingFirmDTO>> getClearingFirms(String token);
        Message<List<TradingFirmDTO>> getTradingFirms(String token);
        Message<String> updateTradingFirmClearingProfile(String token, String clearingFirmId, String tradingFirmId,
                        ProfileDTO profileDTO, String lastUpdateTime);
        Message<String> updateTradingFirmProfile(String token, String tradingFirmId, ProfileDTO profileDTO,
                        String lastUpdateTime);

        Message<List<DistributionListDTO>> getClearingFirmLists(String token, String clearingFirmId);
        Message<List<DistributionListDTO>> getTradingFirmLists(String token, String tradingFirmId);
        Message<DistributionListContentDTO> getDistributionListContent(String token, String id);
        Message<String> updateDistributionListContent(String token, String distributionListId,
                        String content, String lastUpdateTime);
        Message<String> createClearingFirmDistributionList(String token, String clearingFirmId, String name);
        Message<String> createTradingFirmDistributionList(String token, String tradingFirmId, String name);
        Message<String> renameDistributionList(String token, DistributionListDTO distributionListDTO, String lastUpdateTime);
        Message<String> removeDistributionList(String token, String distributionListId, String lastUpdateTime);

        Message<String> shutoffTradingFirmByClearingFirm(String token, String clearingFirmId, String tradingFirmId);
        Message<String> resumeTradingFirmByClearingFirm(String token, String clearingFirmId, String tradingFirmId);
        Message<String> cancelTradingFirmOrdersByClearingFirm(String token, String clearingFirmId, String tradingFirmId);
        Message<String> shutoffAndCancelTradingFirmByClearingFirm(String token, String clearingFirmId, String tradingFirmId);
        Message<String> resetTradingFirmCalculatedNotionalValueByClearingFirm(String token,
                        String clearingFirmId, String tradingFirmId);
        Message<String> shutoffTradingFirm(String token, String tradingFirmId);
        Message<String> resumeTradingFirm(String token, String tradingFirmId);
        Message<String> cancelTradingFirmOrders(String token, String tradingFirmId);
        Message<String> shutoffAndCancelTradingFirm(String token, String tradingFirmId);
        Message<String> resetTradingFirmCalculatedNotionalValue(String token, String tradingFirmId);
}
```

Access to the Kill Switch application is configured using killswitch.properties file which has the following properties:

```
host=devkillswitch.chx.com
port=1422
maxPoolSize=100
```

To use the WebCommon project from another Spring-based application the following line should be included in its Spring Configuration file:

```
<import resource="classpath:WEB-INF/web-service-config.xml"/>
```

Use web service object in Spring controllers:

```
@Autowired
private WebService webService;
```

Object Classes

public class ClearingFirmDTO {

```
	private String id;
	private String name;
	private List<TradingFirmDTO> tradingFirms;
	private List<DistributionListDTO> lists;

	public ClearingFirmDTO() {}

	public ClearingFirmDTO(String id, String name) {
		this.id = id;
		this.name = name;
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public List<TradingFirmDTO> getTradingFirms() {
		return tradingFirms;
	}

	public void setTradingFirms(List<TradingFirmDTO> tradingFirms) {
		this.tradingFirms = tradingFirms;
	}

	public List<DistributionListDTO> getLists() {
		return lists;
	}

	public void setLists(List<DistributionListDTO> lists) {
		this.lists = lists;
	}

}
```

public class TradingFirmDTO {

```
	private String id;
	private String name;
	private ProfileDTO profile;
	private String state;
	private Long currentNv;
	private List<Long> cfMaxValues;
	private List<String> cfNames;
	private List<DistributionListDTO> lists;

	public TradingFirmDTO() {}

	public TradingFirmDTO(String id, String name) {
```

```
                this.id = id;
                this.name = name;
        }

        public String getId() {
                return id;
        }

        public void setId(String id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public ProfileDTO getProfile() {
                return profile;
        }

        public void setProfile(ProfileDTO profile) {
                this.profile = profile;
        }

        public String getState() {
                return state;
        }

        public void setState(String state) {
                this.state = state;
        }

        public Long getCurrentNv() {
                return currentNv;
        }

        public void setCurrentNv(Long currentNv) {
                this.currentNv = currentNv;
        }

        public List<Long> getCfMaxValues() {
                return cfMaxValues;
        }

        public void setCfMaxValues(List<Long> cfMaxValues) {
                this.cfMaxValues = cfMaxValues;
        }

        public List<String> getCfNames() {
                return cfNames;
        }

        public void setCfNames(List<String> cfNames) {
                this.cfNames = cfNames;
        }

        public List<DistributionListDTO> getLists() {
                return lists;
        }

        public void setLists(List<DistributionListDTO> lists) {
                this.lists = lists;
        }

}
```

```
public class DistributionListDTO {

	private String id;
	private String name;

	public DistributionListDTO() {}

	public DistributionListDTO(String id, String name) {
		this.id = id;
		this.name = name;
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}
```

```
public class DistributionListContentDTO {

	private List<String> emails = new ArrayList<String>();

	public DistributionListContentDTO() {}

	public List<String> getEmails() {
		return emails;
	}

	public void setEmails(List<String> emails) {
		this.emails = emails;
	}

	@Override
	public String toString() {
		StringBuilder sb = new StringBuilder();
		if (emails != null) {
			for (String email : emails) {
				sb.append(email);
				sb.append("\n");
			}
		}
		return sb.toString();
	}
}
```

```
public class ProfileDTO {

	private Double maxNv;
	private Integer autoOp;
	private Integer expLimit1;
	private String distListId1;
	private Integer expLimit2;
	private String distListId2;
	private Integer expLimit3;
	private String distListId3;

	public ProfileDTO() {}
```

```
	public Double getMaxNv() {
		return maxNv;
	}

	public void setMaxNv(Double maxNv) {
		this.maxNv = maxNv;
	}

	public Integer getExpLimit1() {
		return expLimit1;
	}

	public void setExpLimit1(Integer expLimit1) {
		this.expLimit1 = expLimit1;
	}
	public String getDistListId1() {
		return distListId1;
	}

	public void setDistListId1(String distListId1) {
		this.distListId1 = distListId1;
	}

	public Integer getExpLimit2() {
		return expLimit2;
	}

	public void setExpLimit2(Integer expLimit2) {
		this.expLimit2 = expLimit2;
	}

	public String getDistListId2() {
		return distListId2;
	}

	public void setDistListId2(String distListId2) {
		this.distListId2 = distListId2;
	}

	public Integer getExpLimit3() {
		return expLimit3;
	}

	public void setExpLimit3(Integer expLimit3) {
		this.expLimit3 = expLimit3;
	}

	public String getDistListId3() {
		return distListId3;
	}

	public void setDistListId3(String distListId3) {
		this.distListId3 = distListId3;
	}

	public Integer getAutoOp() {
		return autoOp;
	}

	public void setAutoOp(Integer autoOp) {
		this.autoOp = autoOp;
	}
}
```

```
public class UserDTO implements Serializable {

	private static final long serialVersionUID = -8718060545014197349L;

	private String fullName;
	private boolean allowAutoAction;
	private String role;
```

```
	private String token;
	private String email;
	private boolean temporaryPassword;
	private boolean locked;
	private String error;

	public String getFullName() {
		return fullName;
	}

	public void setFullName(String fullName) {
		this.fullName = fullName;
	}

	public boolean isAllowAutoAction() {
		return allowAutoAction;
	}
	public void setAllowAutoAction(boolean allowAutoAction) {
		this.allowAutoAction = allowAutoAction;
	}

	public String getRole() {
		return role;
	}

	public void setRole(String role) {
		this.role = role;
	}

	public String getToken() {
		return token;
	}

	public void setToken(String token) {
		this.token = token;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public boolean isTemporaryPassword() {
		return temporaryPassword;
	}

	public void setTemporaryPassword(boolean temporaryPassword) {
		this.temporaryPassword = temporaryPassword;
	}

	public boolean isLocked() {
		return locked;
	}

	public void setLocked(boolean locked) {
		this.locked = locked;
	}

	public String getError() {
		return error;
	}

	public void setError(String error) {
		this.error = error;
	}
}
```

```
public class Message<T> {

        public static final int OK = 200;
        public static final int UNAUTHORIZED = 401;
        public static final int NOT_FOUND = 404;
        public static final int NOT_ACCEPTABLE = 406;
        public static final int PRECONDITION_FAILED = 412;
        public static final int BAD_GATEWAY = 502;
        public static final int SERVICE_UNAVAILBLE = 503;


        private static final HashMap<Integer,String> messages =
                        new HashMap<Integer, String>();

        static {
                messages.put(PRECONDITION_FAILED,
                        "Another user has already updated the profile. Please refresh before update.");
                messages.put(BAD_GATEWAY,
                        "No connection to CHX Gateway. Please try again later.");
                messages.put(SERVICE_UNAVAILBLE,
                         "No connection to CHX Web Service. Please try again later.");
        }

        private int code;
        private String msg;
        private T data;
        private long lastUpdateTime;

        public Message() {}

        public Message (T data) {
                this.code = 200;
                this.msg = "";
                this.data = data;
        }

        public Message(int code) {
                this.code = code;
                this.msg = messages.get(code);
                this.data = null;
        }

        public Message(int code, String msg) {
                this.code = code;
                this.msg = msg;
                this.data = null;
        }

        public int getCode() {
                return code;
        }

        public void setCode(int code) {
                this.code = code;
        }

        public String getMsg() {
                return msg;
        }

        public void setMsg(String msg) {
                this.msg = msg;
        }

        public T getData() {
                return data;
        }

        public void setData(T data) {
                this.data = data;
        }
```

```
	public long getLastUpdateTime() {
		return lastUpdateTime;
	}

	public void setLastUpdateTime(long lastUpdateTime) {
		this.lastUpdateTime = lastUpdateTime;
	}
}
```

```
public class Credentials {

	private String login;
	private String password;

	public Credentials() {}

	public Credentials(String login, String password) {
		this.login = login;
		this.password = password;
	}

	public String getLogin() {
		return login;
	}
	public void setLogin(String login) {
		this.login = login;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}
}
```

Constants

```
public interface UserRole {

        String ADMIN = "admin";
        String CLEARING_FIRM = "clearing_firm";
        String TRADING_FIRM = "trading_firm";

}


public interface TradingFirmState {

        String NORMAL = "active";
        String SHUTOFF = "shutoff";

}


public interface AutoAction {

        int NOTHING = 0;
        int SHUTOFF_FORM = 1;
        int CANCEL_ORDERS = 2;
        int SHUTOFF_AND_CANCEL = 3;

}
```