BA Offline task

Introduction

There are trading Server's entities:

`Symbol`

Field	Description
required **symbol_id**: int64	a financial instrument Id
required **name**: string	a financial instrument name
required interval: Interval[]	trading schedule for a financial instrument - list of intervals when trading is available
...	other symbol parameters

`Interval`

Field	Description
required startSecond: int64	amount of seconds from the Sunday 00:00:00. Determines start of turning Symbol on. For example, 601 means Sunday 00:10:01
required endSecond: int64	amount of seconds from the Sunday 00:00:00. Determines time of turning Symbol off

Current API for Broker Admin Application is implemented as async messages via internal Protocol based on TCP:

Message	Body	Description
GetSymbolListRequest		Get list of existing symbols
GetSymbolListResponse	required symbols: Symbol[]	
CUDSymbolRequest	optional symbol_id: int64 required operation: enum (CREATE, UPDATE, DELETE) ...	Create/Update/Delete symbol
CUDSymbolResponse		Modification is done successfully
ErrorResponse	required code: int32 optional message: string	Error in request
SymbolChangedEvent	required symbol_id: int64 required operation: enum (CREATE, UPDATE, DELETE) ...	Server event on modification

Problem statement

Brokers have many Symbols from different Exchanges (UK, US, etc.) and there is no convenient way to set up a holiday schedule for Symbol In existing implementation. A Holiday is defined a full day when symbol is off and trading is not possible.

Current workaround for Broker is to set up `symbol.interval` for full day in advance and then revert `symbol.interval` back to normal schedule after the holiday.

Task

A ticket needs to be written for the Backend (BE) Team to implement a new feature that resolves the issue described above.

The information in the ticket should be sufficient for developers to implement the feature and for Quality Assurance (QA) to test it.

The ticket should be written in English. Any protocol and approach may be used to complete the task.

Expected structure of ticket:

- **Business value** - This section should answer the question "Why should we do this ticket?" in any format.
- **Description** - This section should include a short description of "How should we implement it?" along with a list of requirements in any format.
- **Use cases** - This is the main section that should include both business and technical details (validations, logic, formulas, etc.).
- **API Protocol** - This section should detail all changes in the API protocol.
- **Test cases** - This section should list test scenarios that need to be covered in Automated Testing (AT) or tested manually and should ensure the quality of the implementation. Test cases should be logically grouped and should not be overly complex. They should be formatted in a table with the following columns:
  1. Test case - A short, human-readable description of the test case.
  2. Role - User (e.g. Trader, Manager), Backend service (e.g. external event), Test environment (e.g. timer)
  3. Action - A description of the action in any format that should be performed by the Role (for example, make a request with parameters).
  4. Result - The expected result that should be triggered by the action and can include several events (Success or Error).