

The Chronobot Bid Manager Documentation

Index

| | |
|--|----|
| Introduction..... | 4 |
| Requirement:..... | 5 |
| Optional: | 5 |
| Installation..... | 6 |
| Setup | 7 |
| Using the ChronoServerWebService web service | 9 |
| WSDL | 9 |
| Protocol Usage and Naming | 11 |
| Request Message:..... | 11 |
| Response Message: | 11 |
| Data Encoding..... | 12 |
| Request Parameter | 12 |
| Request Option Parameter | 12 |
| Response Parameter | 12 |
| Response Option Parameter..... | 12 |
| Response List Parameter..... | 12 |
| Bid Status | 13 |
| Message Definitions..... | 14 |
| (MsgID: 2300) AddUser | 14 |
| (MsgID: 2301) DelUser | 14 |
| (MsgID: 2302) GetUser | 14 |
| (MsgID: 2303) SetUser..... | 15 |
| (MsgID: 2304) UserLogin | 15 |
| (MsgID: 2305) UserLogout | 16 |
| (MsgID: 2400) AddBidRoom..... | 16 |
| (MsgID: 2401) DelBidRoom | 16 |
| (MsgID: 2403) AddCategory..... | 17 |
| (MsgID: 2404) DelCategory | 17 |
| (MsgID: 2405) ListCategory..... | 18 |
| (MsgID: 2406) GetCategory | 18 |
| (MsgID: 2450) ListBidRoom..... | 18 |
| (MsgID: 2453) ListBidRoomBid..... | 19 |
| (MsgID: 2454) StartNewBid..... | 19 |
| (MsgID: 2455) PlaceBid | 20 |
| (MsgID: 2456) StarterUpdateBid | 20 |

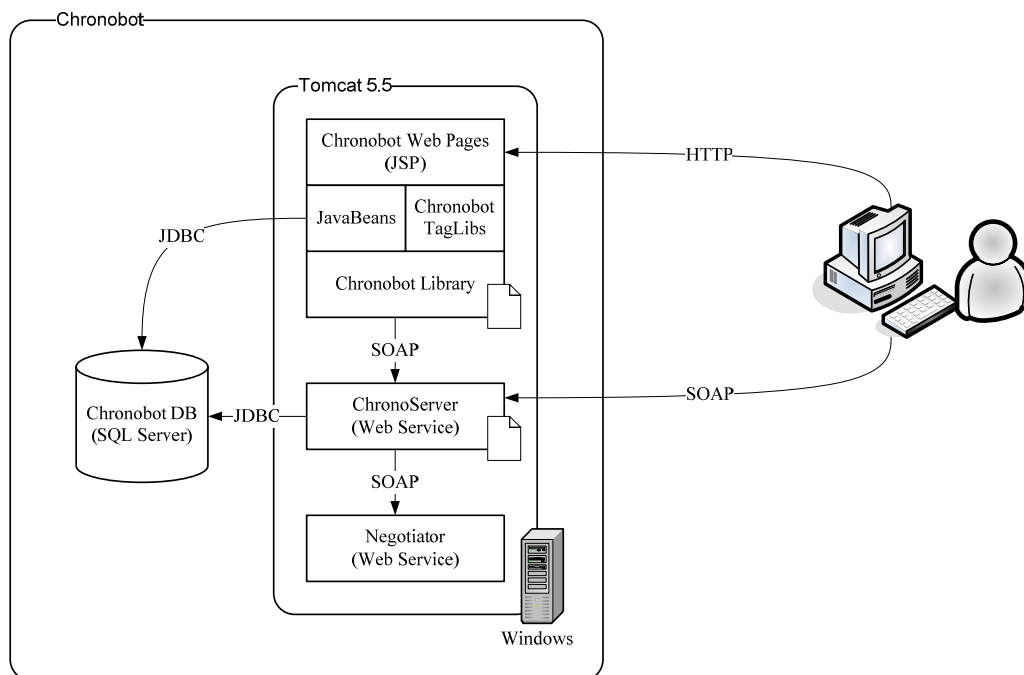
| | |
|---|----|
| (MsgID: 2459) ListStartedBid | 21 |
| (MsgID: 2460) ListPlacedBid..... | 21 |
| (MsgID: 2461) StarterCancelBid..... | 21 |
| (MsgID: 2462) BidderRetractBid | 22 |
| (MsgID: 2463) CancelBid..... | 22 |
| (MsgID: 2464) RetractBid | 22 |
| (MsgID: 2470) QueryBid..... | 23 |
| (MsgID: 2471) ListBidders..... | 23 |
| (MsgID: 2472) PlaceThenCloseBid..... | 24 |
| (MsgID: 2473) CloseBid | 24 |
| (MsgID: 2600) AddTask | 24 |
| (MsgID: 2601) DelTask | 25 |
| (MsgID: 2602) GetTask | 25 |
| (MsgID: 2603) SetTask..... | 25 |
| (MsgID: 2459) ListStartedBid | 26 |
| Database Schema | 27 |
| SQL Command Configuration in <i>web.xml</i> | 31 |
| Implement your closing bid class | 37 |
| Chronobot JSP Tag Library..... | 38 |
| List Tag Result Interface | 38 |
| <chronobot:Attribute> | 39 |
| <chronobot:ListBidRoom> | 39 |
| <chronobot:ListBidRoomBid> | 40 |
| <chronobot:ListPlacedBid> | 40 |
| <chronobot:ListFeasibleBid> | 41 |
| <chronobot:ListStartedBid> | 41 |
| <chronobot:ListTask> | 41 |
| <chronobot:StartNewBid>..... | 42 |
| <chronobot:PlaceBid> | 43 |
| <chronobot:AddUser> | 43 |
| <chronobot:DelUser> | 44 |
| <chronobot:GetUser> | 44 |
| <chronobot:SetUser> | 45 |
| <chronobot:AddBidRoom> | 45 |
| <chronobot:DelBidRoom> | 46 |
| <chronobot:StarterUpdateBid>..... | 46 |
| <chronobot:AddCategory> | 47 |
| <chronobot:AddCategory> | 47 |

| | |
|-------------------------------------|----|
| <chronobot:DelCategory> | 48 |
| <chronobot:ListCategory> | 48 |
| <chronobot:GetCategory> | 49 |
| <chronobot:QueryBid> | 49 |
| <chronobot:ListBidder> | 49 |
| <chronobot:CancelBid> | 50 |
| <chronobot:StarterCancelBid> | 50 |
| <chronobot:RetractBid> | 51 |
| <chronobot:BidderRetractBid> | 51 |
| <chronobot:SignIn> | 52 |
| <chronobot:PlaceThenCloseBid> | 52 |
| <chronobot:CloseBid> | 53 |
| <chronobot:MicroPayment> | 53 |

Introduction

Chronobot Bid Manager provides one web service to facilitate all the bidding process, credits/debits user accounts, and stores bidding transactions into database. This document describes how to install Chronobot Bid Manager. It assumes you already know how to write and run Java code and are not afraid of XML. You should also have an application server, Jakarta Tomcat, and be familiar with operating and deploying to it. If you are installing Tomcat, get the latest 5.5 version. You can use other application server and servlet engines, but you may need to modify the *web.xml* and some source code, because we use the [Apache Jakarta Tomcat 5's JNDI InitialContext implementation](#) and JNDI Environment Properties. Note also that the Chronobot Bid Manager requires Java 1.4 or later.

What we provide are three web applications, which are **ChronoServerWebService**, **ChronobotNegotiation**, and **Chronobot**. ChronoServerWebService is the web service version of Bid Manager. ChronobotNegotiation is a dummy negotiator, which just return a random result. You may write your own negotiator web service and make Bid Manager call that web service. The last one, Chronobot, is many web page samples which use taglibs and JavaBeans to call the Bid Manager. Those web pages are used in another scenario and are using Traditional Chinese with Big-5 encoding, not English with UTF-8 encoding. But you may find the taglibs are very useful if you are trying to create some web pages to call the ChronoServerWebService.



Requirement:

- Java SDK: J2SE 1.4.2 SDK or later, [J2SE 5.0](#) is recommended.
- Tomcat 5 or later, [Tomcat 5.5](#) is recommended.
- Database: Microsoft SQL Server 2000 is recommended. (You can use other database, but you may need to modify some SQL command in *web.xml*)
- JDBC Driver: [Microsoft SQL Server 2000 Driver for JDBC](#). (If you use other database, then use that database's JDBC driver.)

Optional:

For AXIS only, and not used in The Chronobot Bid Manager yet.

- JavaMail: [JavaMail 1.3.3](#).
- JavaBeans Activation Framework: [JAF 1.0.2](#).

If you need to modify the source code and rebuild it, then you may need some tools to help you.

- Java build tool: [Ant 1.6.5](#) or [NetBeans 4.1](#)

The AXIS 1.3 lib*.jar files are included in the WAR files. If you need to update or change for any reason, here is for your reference.

- Java Web Service: [AXIS 1.3 Final](#).

Installation

After installing the Java SDK, Tomcat, and Microsoft SQL Server 2000, you need to copy those Shared Library Files into Tomcat 5:

- Copy JDBC drivers that are required for both your application and internal Tomcat use into ***\$CATALINA_HOME\common\lib***. If you are using Microsoft SQL Server 2000, just copy *\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib\[msbase|mssqlserver|msutil].jar* to ***\$CATALINA_HOME\common\lib***.

- Copy other shared library files into ***\$CATALINA_HOME\shared\lib***. In this case, we just copy *activation.jar* and *mail.jar* into ***\$CATALINA_HOME\shared\lib***. The reason we do so is just because AXIS will check these two libraries and cause a warning messages if they are not found.

- Deploy three WAR files into ***\$CATALINA_HOME\webapps***, you may just copy three WAR files into that folder, or copy unpacked directory hierarchy into a subdirectory in directory ***\$CATALINA_HOME\webapps***. About Tomcat 5 deployment, [here is for your reference](#). Be sure to restart Tomcat after installing or updating your application.

Setup

After above steps, you need to modify the configuration file:

■ [Chronobot|ChronoServerWebService]\META-INF\context.xml:

Setup the database connection pool information.

We use JNDI naming service to setup the JDBC Connection Pool.

This example setting is for Microsoft SQL Server 2000:

```
<Resource auth="Container"
driverClassName="com.microsoft.jdbc.sqlserver.SQLServerDriver"
logAbandoned="true" maxActive="8" maxIdle="4" name="jdbc/Chronobot"
username="Chronobot" password="Chr0n0b0t" removeAbandoned="true"
removeAbandonedTimeout="60" type="javax.sql.DataSource"
url="jdbc:microsoft:sqlserver://localhost:1433;databasename=ChronoD
B"/>
```

- driverClassName: The Driver for JDBC.
- maxActive: The maximum number of connections in the pool in use at any one time. The pool will not grow beyond this size.
- maxIdle: The maximum number of idle connections in the pool at one time. The pool will shrink in size if there are more than this number of idle connections.
- username & password: The database valid user name and its password.
- name: This resource name must match with the setting in *web.xml*.
- url: Database connection URL. Reference to your JDBC driver document.

You must change the url property to fit your real database connection and database name. You also need to update username and password properties.

■ ChronoServerWebService\WEB-INF\web.xml:

The most important configuration file. Right now, we just focus on the database connection pool setting.

This example setting is for Microsoft SQL Server 2000:

```
<resource-ref>
  <description>Chronobot Database Connection Pool</description>
  <res-ref-name>jdbc/Chronobot</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
<env-entry>
```

```

    <env-entry-name>NegoServer</env-entry-name>
    <env-entry-value>
http://localhost:8080/ChronobotNegotiation/services/ChronobotNegoti
ation
    </env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>

```

- `res-ref-name`: Resource name, must match with the setting in *context.xml*.

NegoServer is setting the Negotiation Web Service's URL, you may need to update it if you change the port or hostname of Tomcat 5.

■ ***Chronobot\WEB-INF\web.xml:***

Chronobot folder contains many web sample pages which use Taglibs or JavaBeans to call ChronoServerWebService.

```

<env-entry>
    <env-entry-name>ChronoServer</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>
http://localhost:8080/ChronoServerWebService/services/chronobot
    </env-entry-value>
</env-entry>

```

ChronoServer is setting the ChronoServerWebService's URL, you may need to update it if you change the port or hostname of Tomcat 5.

Using the ChronoServerWebService web service

WSDL

Basically, this web service accepts two Vector parameters, which stand for key and value pairs. For example, the login message elements in Key Vector are “MsgID”, “UserId”, and “UserPwd”, and the elements in Value Vector are “2304.1”, “Willy”, “123123”, then the input means: MsgID=”2304.1”, UserId=”Willy” and UserPwd=”123123”. The web service returns one hashtable. For example, {{"MsgID", "2453.2"}, {"UserId", "Willy"}, {"BidRoomId", "1"}, {"bid_id.1", "23"}, {"bidroom_id.1", "1"}, {"category.1", "53"}, ..., {"bid_id.2", "24"}, {"bidroom_id.2", "1"}, {"category.2", "53"}, ... {"bid_id.3", "25"}, {"bidroom_id.3", "1"}, {"category.3", "51"}, ...}. This returned hashtable contains a three row *ResultSet*, which are:

| | bid_id, | bidroom_id, | category, | ... |
|----|---------|-------------|-----------|-----|
| 1. | 23, | 1, | 53, | ... |
| 2. | 24, | 1, | 53, | ... |
| 3. | 25, | 1, | 51, | ... |

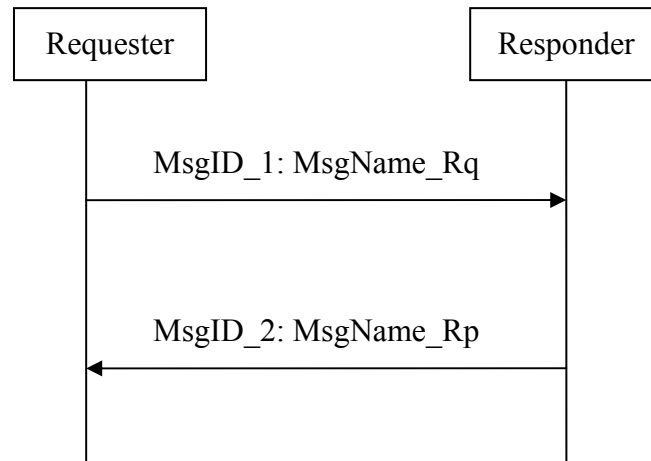
Below is the web service WSDL file.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://ChronoServer"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apacheSOAP="http://xml.apache.org/xml-soap" xmlns:impl="http://ChronoServer"
xmlns:intf="http://ChronoServer"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
xmlns:wSDLSOAP="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xSD="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <schema targetNamespace="http://xml.apache.org/xml-soap"
xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="Vector">
        <sequence>
          <element maxOccurs="unbounded" minOccurs="0" name="item" type="xsd:anyType"/>
        </sequence>
      </complexType>
      <complexType name="mapItem">
        <sequence>
          <element name="key" nillable="true" type="xsd:string"/>
          <element name="value" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="Map">
        <sequence>
          <element maxOccurs="unbounded" minOccurs="0" name="item" type="apacheSOAP:mapItem"/>
        </sequence>
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="processResponse">
    <wsdl:part name="processReturn" type="apacheSOAP:Map"/>
  </wsdl:message>
  <wsdl:message name="processRequest">
    <wsdl:part name="in0" type="apacheSOAP:Vector"/>
    <wsdl:part name="in1" type="apacheSOAP:Vector"/>
  </wsdl:message>
  <wsdl:portType name="ChronoWSEntry">
    <wsdl:operation name="process" parameterOrder="in0 in1">
      <wsdl:input message="impl:processRequest" name="processRequest"/>
      <wsdl:output message="impl:processResponse" name="processResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ChronoServerSoapBinding" type="impl:ChronoWSEntry">
    <wsdl:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  </wsdl:binding>
</wsdl:definitions>
```

```
<wsdl:operation name="process">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="processRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://ChronoServer" use="encoded"/>
  </wsdl:input>
  <wsdl:output name="processResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://ChronoServer" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ChronoWSEntryService">
  <wsdl:port binding="impl:ChronoServerSoapBinding" name="ChronoServer">
    <wsdlsoap:address
location="http://127.0.0.1:8080/BidMgrWebService/services/ChronoServer"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Protocol Usage and Naming

Two types of messages, called *Request* and *Response*, are used for service interactions. Typical use of protocol is shown as below, Requester sends a request message to Responder and then it returns results in a response message.



The general rules of message naming are listed below. Message Identifier (MsgId) and Name (MsgName) is given in Message Definitions Section. In those definitions, request/response parameters compose the body of request/response messages herein.

Request Message:

- Number: MsgId_1 (e.g., 2300.1)
- Name: MsgName_Rq (e.g., AddUser_Rq)

Response Message:

- Number: MsgId_2 (e.g., 2300.2)
- Name: MsgName_Rp (e.g., AddUser_Rp)

Data Encoding

We use a data encoding naming method to allow optional request and response parameter.

Request Parameter

A request parameter is named **without** *‘.opt’* postfix. The normal request parameter is needed by the BidManager. For example, start new bid message’s request parameters are: *UserId, UserPwd, BidRoomId, BidDuration, EndTime*.

Request Option Parameter

A request option parameter is named **with** *‘.opt’* postfix. The request Option parameter does not appear in BidManager’s code; but it will be added into database’s table dynamically. As mentioned as above, start new bid message’s request parameters are: *UserId, UserPwd, BidRoomId, BidDuration, EndTime*. Others are option parameters, for example, a *‘bid_descr.opt’* parameter is mapping to the *‘bid_descr’* column in the table.

Response Parameter

A response parameter is named **without** *‘.i’* postfix, where $i \in \mathbb{N}$. The normal response parameter name is coded in the BidManager’s code. For example, start new bid message’s response parameters are: *UserId, BidRoomId, Success, Reason, BidId, Status, start_time, end_time*.

Response Option Parameter

A response option parameter is named **without** *‘.i’* postfix, where $i \in \mathbb{N}$. The response option parameter name is not coded in the BidManager’s code; but it can be added into response parameters from SQL query result dynamically.

Response List Parameter

A response list parameter is named **with** *‘.i’* postfix, where $i \in \mathbb{N}$. The response list parameter name is not coded in the BidManager’s code; and it can be added into response parameters from SQL query result dynamically. For example, $\text{CategoryList}\{r_i\}$, $i \in \mathbb{N}$: List of categories. $r_i = \langle \text{CategoryId}, \text{CategoryName}, \text{CategoryNamePath}, \text{BidNum} \rangle$. Then the response list parameters are: $\{\{\text{“cat_id.1”}, \text{“34”}\}, \{\text{“cat_name.1”}, \text{“nurse”}\}, \{\text{“name_path.1”}, \text{“/healthcare”}\}, \{\text{“BidNum.1”}, \text{“4”}\}, \{\text{“cat_id.2”}, \text{“35”}\}, \{\text{“cat_name.2”}, \text{“elder”}\}, \{\text{“name_path.2”}, \text{“/healthcare”}\}, \{\text{“BidNum.2”}, \text{“6”}\}, \dots\}$

Bid Status

The status of a bid must be one of the seven defined states below.

States:

- 0: Open: the bid is active.
- 1: Failed: the bid is closed but its goal is not fulfilled.
- 2: Partial closed: the bid is closed and its goal is partial fulfilled.
- 3: Closed: the bid is closed and its goal is all fulfilled.
- 4: Negotiating: the bid is waiting the negotiation result.
- 5: Failed negotiated: the bid is closed by negotiation, but its goal is not fulfilled.
- 6: Partial negotiated: the bid is closed by negotiation, and its goal is partial fulfilled.
- 7: Negotiated: the bid is closed by negotiation, and its goal is all fulfilled.

Message Definitions

(MsgID: 2300) AddUser

User requests BM to accept a new registration. BM returns whether the registration was successful.

Request Parameters:

- UserId: This id is required in the case of the administrator adds a new user.
- UserPwd: Administrator's password
- TgtUserId: target user to be added.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: True or False. Whether the request was successful.
- Reason: Failure reasons.

(MsgID: 2301) DelUser

User requests BM to delete the given user. BM returns whether the deletion was successful.

Request Parameters:

- UserId: If UserId is not equal to TgtUserId, BM must check whether the requestor has sufficient permissions.
- UserPwd
- TgtUserId: target user to be deleted.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: True or False. Whether the request was successful.
- Reason: Failure reasons.

(MsgID: 2302) GetUser

User requests BM to show user profile of the given user id. BM returns complete user information.

Request Parameters:

- ReqUserId: If ReqUserId is not equal to TgtUserId, BM must check whether the requestor has sufficient permissions.
- ReqUserPwd

- TgtUserId: Id of target user whose profile is to be retrieved.
- *Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: True or False. Whether the request was successful.
- Reason: Failure reasons.
- *Options*: Response option parameters.

(MsgID: 2303) SetUser

User requests BM to update a profile. Users can update their personal data and administrators may change profiles of other people. BM returns the updated profile of the given user, just as the GetUser message does.

Request Parameters:

- ReqUserId: If ReqUserId is not equal to TgtUserId, BM checks whether the requestor has sufficient permissions.
- ReqUserPwd
- TgtUserId: Id of target user whose profile is to be updated.
- *Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: True or False. Whether the request was successful.
- Reason: Failure reasons.

(MsgID: 2304) UserLogin

User requests BM to accept his/her entry of bid system.

Request Parameters:

- UserId: user id
- UserPwd: User password.
- *Option.opt*: Request option parameters.

Response Parameters:

- Success: True or False. Whether the login was successful.
- Reason: Failure reasons.
- LoginMsg: System message (such as welcome or announcement) for user login
- *Options*: Response option parameters.

(MsgID: 2305) UserLogout

User requests BM to accept his/her exit of bid system.

Request Parameters:

- UserId
- UserPwd

Response Parameters:

- UserId
- Success: True or False. Whether the logout was successful
- Reason: failure reasons.
- LogoutMsg: system message for user logout.

(MsgID: 2400) AddBidRoom

User (administrator) requests BM to create a bid room. BM will allocate a unique id for this room and return the id to user if everything was successful. Failure reasons of creation are also returned.

Request Parameters:

- UserId
- UserPwd
- BidRoomName: Name of the new bid room. Has to be unique else the request will be rejected.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId
- BidRoomId: Id of the added bid room. Null if creation is failed.
- BidRoomName: Name of the new bid room. Has to be unique else the request will be rejected.
- Success: True or False. If the creation of bid room was successful or not.
- Reason: If there is a failure, the reason for it. Duplicate bid room name and data operations failed can be the reasons for rejection.

(MsgID: 2401) DelBidRoom

User (administrator) requests BM to delete a bid room. The failure can be either the bid room id does not exist or there are active bids in the bid room.

Request Parameters:

- UserId
- UserPwd

- BidRoomId: Id of the bid room that is to be deleted.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId
- Success: True or False. Whether the deletion of this bid room was successful or not.
- Reason: Failure reasons. Active bids in the bid room or bid room id not existing can be the reasons.

(MsgID: 2403) AddCategory

User (administrator) requests BM to add a new category in the bid room.

Request Parameters:

- UserId
- UserPwd
- BidRoomId: Id of the bid room in which to add the category.
- ParentCatId: Where this category is located.
- CategoryName: The category's name.
- LinkCategoryId: The linked category's id, if any.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId
- CategoryId: Id of the new category.
- Success: True or False. Whether the creation of this category was successful or not.
- Reason: Failure reasons.

(MsgID: 2404) DelCategory

User (administrator) requests BM to del a category in the bid room.

Request Parameters:

- UserId
- UserPwd
- BidRoomId: Id of the bid room in which to del the category.
- CategoryId: Id of the category.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId

- Success: True or False. Whether the creation of this category was successful or not.
- Reason: Failure reasons.

(MsgID: 2405) ListCategory

User asks BM to list all the sub-category in current bid room and current category. BM returns the list of categories.

Request Parameters:

- UserId
- UserPwd
- BidRoomId: Id of the bid room in which to list the category.
- CategoryId: Id of the current category.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId
- OptionList*: Response list parameters.

(MsgID: 2406) GetCategory

User asks BM to show category of the given category id. BM returns complete category information.

Request Parameters:

- UserId
- UserPwd
- BidRoomId: Id of the bid room in which to list the category.
- CategoryId: Id of the current category.

Response Parameters:

- UserId
- Options*: Response option parameters.

(MsgID: 2450) ListBidRoom

User asks BM to list all the active bid rooms. BM returns the list of active bid rooms.

Request Parameters:

- UserId
- UserPwd
- Option.opt*: Request option parameters.

Response Parameters:

- UserId
- OptionList*: Response list parameters.

(MsgID: 2453) ListBidRoomBid

User requests BM to list bids in the bid room.

Request Parameters:

- UserId
- UserPwd
- BidRoomId: Id of the bid room in which to list the users.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId
- BidRoomId: Id of the bid room for which to list all the bids.
- OptionList*: Response list parameters.

(MsgID: 2454) StartNewBid

User requests BM to start a new bid on his/her behalf. Properties of the bid are given below. BM returns if the request was successful. In case of a failure, reason for the failure (of start of the bid) must be specified in response message. On Success, the optional properties of the response message describe the result of the active bid. Moreover, BM returns a unique id for this bid (it's unique per bid room). Any messages from the user pertaining to this bid should carry the bid id and bid room id along with other parameters.

Request Parameters:

- UserId: User id of the person wanting to start a new bid.
- UserPwd
- BidRoomId: Id of the bid room where the bid is to be started.
- NumHours: Number of hours requested in the bid.
- BidDuration: How long will the bidding process continue.
- EndTime: If BidDuration is not specified, then EndTime take effect.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- BidRoomId: Id of the bid room in question
- Success: True or False. Whether the user request was successful.

- Reason: If there is a failure, why the request failed.
- BidId: Id of the opened bid.
- Status: bid status.
- start_time: When the task start
- end_time: When the task end

(MsgID: 2455) PlaceBid

User tells BM that s/he wants to place a bid for the given bid id. BM responds to the bidder if the bid was successful. Failure reasons are returned if request failed.

Request Parameters:

- UserId: User id of the person bidding
- UserPwd
- BidRoomId: Id of the bid room where the person is bidding
- BidId: Bid id on which the person is bidding
- AllowMultiBid: Allow or not one user places the same bid more than once.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- BidRoomId: Id of the bid room where the bid is active.
- BidId: Bid id of the bid for which this message is.
- Success: True or False. Whether the request was successful.
- Reason: Failure reasons.

(MsgID: 2456) StarterUpdateBid

Bid starter requests BM to update parameter values of his/her bid. Changeable parameters of an active bid are listed as request option parameters.

Request Parameters:

- UserId: User id of the person wanting to start a new bid.
- UserPwd
- BidRoomId: Id of the bid room where the bid is to be started.
- BidId: Id of the bid.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: True or False. Whether the user request was successful or not.
- Reason: If there is a failure, why the request failed. The bid was closed can be

a reason.

(MsgID: 2459) ListStartedBid

User requests BM to list the bids s/he started.

Request Parameters:

- UserId: User id.
- UserPwd
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- OptionList*: Response list parameters.

(MsgID: 2460) ListPlacedBid

User requests BM to list the bids s/he has been placed.

Request Parameters:

- UserId: User id.
- UserPwd
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- OptionList*: Response list parameters.

(MsgID: 2461) StarterCancelBid

The bid starter tells BM s/he wants to cancel the bid. BM returns whether the canceling was successful.

Request Parameters:

- UserId: User id of the person canceling.
- UserPwd
- BidRoomId: Id of the bid room which contains the bid to be canceled.
- BidId: Id of bid to be canceled.
- Reason: Why the starter cancel the bid.
- Option.opt*: Request option parameters.

Response Parameters:

- Success: True or False. Whether the user request was successful or not.
- Reason: If there is a failure, why the request failed.

(MsgID: 2462) BidderRetractBid

The bidder tells BM that s/he want to retract the bid. BM returns whether the retracting was successful.

Request Parameters:

- UserId: User id of the person withdrawing.
- UserPwd
- BidRoomId: Id of the bid room which contains the target bid.
- BidId: Bid id on which the person is withdrawing.
- Reason: why the bidder retract the bid.
- Option.opt*: Request option parameters.

Response Parameters:

- Success: True or False. Whether the user request was successful or not.
- Reason: If there is a failure, why the request failed. No appropriate substitute can be a reason.

(MsgID: 2463) CancelBid

The administrator tells BM s/he wants to cancel the bid. BM returns whether the canceling was successful.

Request Parameters:

- UserId: User id of the administrator canceling.
- UserPwd
- StarterId: User id of the person starting the bid.
- BidRoomId: Id of the bid room which contains the bid to be canceled.
- BidId: Id of bid to be canceled.
- Reason: Why admin cancel the bid.
- Option.opt*: Request option parameters.

Response Parameters:

- Success: True or False. Whether the user request was successful or not.
- Reason: If there is a failure, why the request failed.

(MsgID: 2464) RetractBid

The administrator tells BM that s/he want to retract the bid. BM returns whether the retracting was successful.

Request Parameters:

- UserId: User id of the administrator withdrawing.

- UserPwd
- BidderId: User id of the person bidding the bid.
- BidRoomId: Id of the bid room which contains the target bid.
- BidId: Bid id on which the person is withdrawing.
- Reason: why the bidder retract the bid.
- Option.opt*: Request option parameters.

Response Parameters:

- Success: True or False. Whether the user request was successful or not.
- Reason: If there is a failure, why the request failed. No appropriate substitute can be a reason.

(MsgID: 2470) QueryBid

User requests BM to get a bid's open information.

Request Parameters:

- UserId: User id of the person wanting to query a bid.
- UserPwd
- BidId: Id of the bid to be queried.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: True or False. Whether the query was successful or not.
- Reason: If there is a failure, why the request failed.
- OptionList*: Response list parameters.

(MsgID: 2471) ListBidders

User requests BM to get a bid's bidders.

Request Parameters:

- UserId: User id of the person wanting to query a bid's bidders.
- UserPwd
- BidId: Id of the bid to be queried.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- OptionList*: Response list parameters.

(MsgID: 2472) PlaceThenCloseBid

User tells BM that s/he wants to place a bid for the given bid id and then close it. BM responds to the bidder if the request was successful. Failure reasons are returned if request failed.

Request Parameters:

- UserId: User id of the person bidding
- UserPwd
- BidRoomId: Id of the bid room where the person is bidding
- BidId: Bid id on which the person is bidding
- AllowMultiBid: Allow or not one user places the same bid more than once.
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- BidRoomId: Id of the bid room where the bid is active.
- BidId: Bid id of the bid for which this message is.
- Success: True or False. Whether the request was successful.
- Reason: Failure reasons.

(MsgID: 2473) CloseBid

The administrator tells BM s/he wants to close the bid. BM returns whether the request was successful.

Request Parameters:

- UserId: User id of the administrator canceling.
- UserPwd
- BidRoomId: Id of the bid room which contains the bid to be canceled.
- BidId: Id of bid to be canceled.
- Option.opt*: Request option parameters.

Response Parameters:

- Success: True or False. Whether the user request was successful or not.
- Reason: If there is a failure, why the request failed.

(MsgID: 2600) AddTask

User requests TKM to add a new task in his/her time schedule. TKM returns if the addition was successful.

Request Parameters:

- UserId: user identifier

- UserPwd: user password
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- Success: true/false. Whether this request was successful or not.
- Reason: if there is a failure, why the request failed.

(MsgID: 2601) DelTask

User requests TKM to delete the given task in his/her time schedule. TKM returns if the deletion was successful.

Request Parameters:

- UserId: user identifier
- UserPwd: user password
- TaskId: task id

Response Parameters:

- UserId: User id for whom the result is.
- Success: ture/false. Whether this request was successful or not.
- Reason: if there is a failure, why the request failed.

(MsgID: 2602) GetTask

User requests TKM to return all information of the given task in his/her time schedule.

Request Parameters:

- UserId: user identifier
- UserPwd: user password
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: user identifier
- OptionList*: Response list parameters.

(MsgID: 2603) SetTask

User requests TKM to update parameter values of the given task in his/her time schedule. Note that all of the parameters are optional, but to have any effect at least one of them must be present. TKM returns the complete information of the task, just as the GetTask message does, which will include any updated values.

Parameters:

- UserId: user identifier
- UserPwd: user password
- TaskId
- TaskDescription: detail descriptions of the task
- StartTime: start time of the task
- EndTime: end time of the task
- Location: physical or virtual location

Response Parameters:

- Success: true/false. Whether this request was successful or not.
- Reason: if there is a failure, why the request failed.

(MsgID: 2459) ListStartedBid

The user requests TKM to list all feasible bids according to his schedule (time) and capability (knowledge).

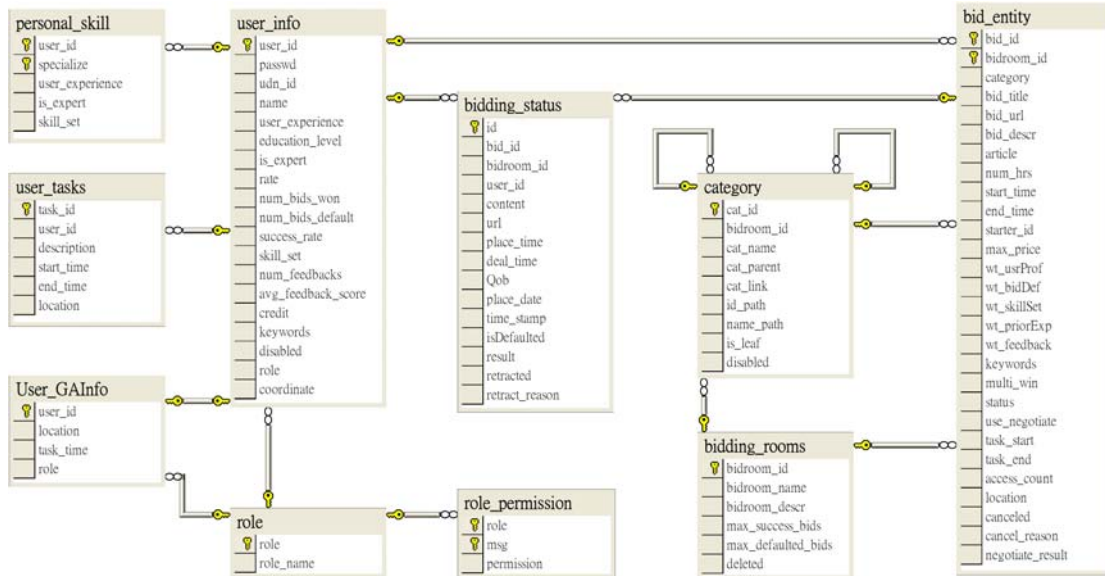
Request Parameters:

- UserId: User id.
- UserPwd
- Option.opt*: Request option parameters.

Response Parameters:

- UserId: User id for whom the result is.
- OptionList*: Response list parameters.

Database Schema



Above is the database schema, and all of the table names and column names are changeable. You can add, modify, or delete any column in any table, but you may need to change the SQL command in *ChronoServerWebService/WEB-INF/web.xml*. Below is sql commands to create our sample database which is matching with the sample version configuration file, *ChronoServerWebService/WEB-INF/web.xml*.

```

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_bidding_status_bid_entity]') and OBJECTPROPERTY(id,
N'IsForeignKey') = 1)
ALTER TABLE [dbo].[bidding_status] DROP CONSTRAINT FK_bidding_status_bid_entity
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_bid_entity_bidding_rooms]') and OBJECTPROPERTY(id, N'IsForeignKey')
= 1)
ALTER TABLE [dbo].[bid_entity] DROP CONSTRAINT FK_bid_entity_bidding_rooms
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_category_bidding_rooms]') and OBJECTPROPERTY(id, N'IsForeignKey')
= 1)
ALTER TABLE [dbo].[category] DROP CONSTRAINT FK_category_bidding_rooms
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_bid_entity_category]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[bid_entity] DROP CONSTRAINT FK_bid_entity_category
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_category_category]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[category] DROP CONSTRAINT FK_category_category
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_category_category1]') and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[category] DROP CONSTRAINT FK_category_category1
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_role_permission_role]') and OBJECTPROPERTY(id, N'IsForeignKey') =
1)
ALTER TABLE [dbo].[role_permission] DROP CONSTRAINT FK_role_permission_role
GO

```

```

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_User_GAInfo_role]')
and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[User_GAInfo] DROP CONSTRAINT FK_User_GAInfo_role
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[FK_user_info_role]')
and OBJECTPROPERTY(id, N'IsForeignKey') = 1)
ALTER TABLE [dbo].[user_info] DROP CONSTRAINT FK_user_info_role
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_bid_entity_user_info]') and OBJECTPROPERTY(id, N'IsForeignKey') =
1)
ALTER TABLE [dbo].[bid_entity] DROP CONSTRAINT FK_bid_entity_user_info
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_bidding_status_user_info]') and OBJECTPROPERTY(id, N'IsForeignKey')
= 1)
ALTER TABLE [dbo].[bidding_status] DROP CONSTRAINT FK_bidding_status_user_info
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_personal_skill_user_info]') and OBJECTPROPERTY(id, N'IsForeignKey')
= 1)
ALTER TABLE [dbo].[personal_skill] DROP CONSTRAINT FK_personal_skill_user_info
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_User_GAInfo_user_info]') and OBJECTPROPERTY(id, N'IsForeignKey') =
1)
ALTER TABLE [dbo].[User_GAInfo] DROP CONSTRAINT FK_User_GAInfo_user_info
GO

if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[FK_user_tasks_user_info]') and OBJECTPROPERTY(id, N'IsForeignKey') =
1)
ALTER TABLE [dbo].[user_tasks] DROP CONSTRAINT FK_user_tasks_user_info
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[User_GAInfo]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[User_GAInfo]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[bid_entity]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[bid_entity]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[bidding_rooms]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[bidding_rooms]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[bidding_status]')
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[bidding_status]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[category]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[category]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[role]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[role]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[role_permission]')
and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[role_permission]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[user_info]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[user_info]
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[user_tasks]') and
OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[user_tasks]
GO

CREATE TABLE [dbo].[User_GAInfo] (
[user_id] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,

```

```

        [location] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
        [task_time] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
        [role] [tinyint] NULL
    ) ON [PRIMARY]
GO

CREATE TABLE [dbo].[bid_entity] (
    [bid_id] [int] IDENTITY (1, 1) NOT FOR REPLICATION NOT NULL ,
    [bidroom_id] [int] NOT NULL ,
    [category] [int] NULL ,
    [bid_title] [varchar] (100) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [bid_url] [varchar] (200) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [bid_descr] [varchar] (2000) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [article] [ntext] COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [num_hrs] [tinyint] NULL ,
    [start_time] [datetime] NOT NULL ,
    [end_time] [datetime] NULL ,
    [starter_id] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [max_price] [int] NULL ,
    [wt_usrProf] [tinyint] NULL ,
    [wt_bidDef] [tinyint] NULL ,
    [wt_skillSet] [tinyint] NULL ,
    [wt_priorExp] [tinyint] NULL ,
    [wt_feedback] [tinyint] NULL ,
    [keywords] [varchar] (60) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [multi_win] [tinyint] NULL ,
    [status] [tinyint] NOT NULL ,
    [use_negotiate] [tinyint] NULL ,
    [task_start] [datetime] NULL ,
    [task_end] [datetime] NULL ,
    [access_count] [int] NULL ,
    [location] [varchar] (100) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [canceled] [tinyint] NOT NULL ,
    [cancel_reason] [varchar] (500) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [negotiate_result] [varchar] (200) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

CREATE TABLE [dbo].[bidding_rooms] (
    [bidroom_id] [int] IDENTITY (1, 1) NOT FOR REPLICATION NOT NULL ,
    [bidroom_name] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [bidroom_descr] [varchar] (200) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [max_success_bids] [int] NOT NULL ,
    [max_defaulted_bids] [int] NOT NULL ,
    [deleted] [tinyint] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[bidding_status] (
    [id] [int] IDENTITY (1, 1) NOT FOR REPLICATION NOT NULL ,
    [bid_id] [int] NOT NULL ,
    [bidroom_id] [int] NOT NULL ,
    [user_id] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [content] [varchar] (2000) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [url] [varchar] (200) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [place_time] [tinyint] NULL ,
    [deal_time] [tinyint] NULL ,
    [Qob] [int] NULL ,
    [place_date] [datetime] NULL ,
    [time_stamp] [timestamp] NOT NULL ,
    [isDefaulted] [tinyint] NOT NULL ,
    [result] [tinyint] NOT NULL ,
    [retracted] [tinyint] NOT NULL ,
    [retract_reason] [varchar] (200) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[category] (
    [cat_id] [int] IDENTITY (1, 1) NOT FOR REPLICATION NOT NULL ,
    [bidroom_id] [int] NOT NULL ,
    [cat_name] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [cat_parent] [int] NOT NULL ,
    [cat_link] [int] NOT NULL ,
    [id_path] [varchar] (30) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [name_path] [varchar] (200) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [is_leaf] [int] NOT NULL ,
    [disabled] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[role] (
    [role] [tinyint] IDENTITY (1, 1) NOT NULL ,
    [role_name] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[role_permission] (

```

```

        [role] [tinyint] NOT NULL ,
        [msg] [int] NOT NULL ,
        [permission] [int] NOT NULL
    ) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[user_info] (
    [user_id] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [passwd] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [udn_id] [int] NULL ,
    [name] [varchar] (80) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [user_experience] [int] NULL ,
    [education_level] [varchar] (50) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [is_expert] [tinyint] NULL ,
    [rate] [int] NULL ,
    [num_bids_won] [int] NOT NULL ,
    [num_bids_default] [int] NOT NULL ,
    [success_rate] [int] NOT NULL ,
    [skill_set] [int] NOT NULL ,
    [num_feedbacks] [int] NOT NULL ,
    [avg_feedback_score] [int] NOT NULL ,
    [credit] [int] NULL ,
    [keywords] [varchar] (60) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL ,
    [disabled] [tinyint] NOT NULL ,
    [role] [tinyint] NOT NULL ,
    [coordinate] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL
) ON [PRIMARY]
GO

```

```

CREATE TABLE [dbo].[user_tasks] (
    [task_id] [int] IDENTITY (1, 1) NOT FOR REPLICATION NOT NULL ,
    [user_id] [varchar] (20) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [description] [varchar] (250) COLLATE Chinese_Taiwan_Stroke_CI_AS NOT NULL ,
    [start_time] [datetime] NOT NULL ,
    [end_time] [datetime] NOT NULL ,
    [location] [varchar] (100) COLLATE Chinese_Taiwan_Stroke_CI_AS NULL
) ON [PRIMARY]
GO

```

SQL Command Configuration in *web.xml*.

ChronoServerWebService/WEB-INF/web.xml is the most important configuration file. We use Tomcat 5's JNDI Environment Properties to store every SQL command in the Bid Manager. So that you can change the database schema, table names, and column names without recompiling the source code. The Bid Manager Web Service accepts request option parameters which are combined with **column names** and *.opt* postfix. For example, the Bid Manager Web Service received a message as follow:

Key Vector: {"MsgID", "UserId", "UserPwd", "BidRoomId", "category.opt", "status.opt"}

Value Vector: {"2453.1", "Willy", "123123", "2", "25", "0"}

For SQL SELECT command, these request option parameters will be transformed as *optionConditionList*: **"AND category = '25' AND status = '0'"**

For SQL INSERT command, these request option parameters will be transformed as *optionNameList*: **“, category, status”** and *optionValueList*: **“, '25', '0'”**.

For SQL UPDATE command, these request option parameters will be transformed as *optionValueList*: **“, category = '25', status = '0'”**.

Because “MsgID: 2453.1” is ListBidRoomBid message, and the corresponding SQL command setting is:

```
<env-entry>
  <env-entry-name>listBidRoomBids</env-entry-name>
  <!-- 0:bidroom_id 1:optionConditionList -->
  <env-entry-value>
    SELECT * FROM bid_entity WHERE canceled=0 AND bidroom_id={0}{1}
  </env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

And then Bid Manager will transform the SQL command to:

```
SELECT * FROM bid_entity WHERE canceled=0 AND bidroom_id=2 AND category='25' AND status='0'
```

Where {0} is replaced with “2”, and {1} is replaced with *optionConditionList*, which is **"AND category = '25' AND status = '0'"**.

Let's see another sample, AddTask (MsgID: 2600):

Key Vector: {"MsgID", "UserId", "UserPwd", "description.opt", "start_time.opt", "end_time.opt", "location.opt"}

Value Vector: {"2600.1", "Willy", "123123", "Meeting", "2006-1-16 10:00:00", "2006-1-16 12:00:00", "Pittsburg"}

The corresponding SQL command setting is:

```
<env-entry>
  <env-entry-name>addTask</env-entry-name>
  <!-- 0:user_id 1:optionNameList 2:optionValueList -->
  <env-entry-value>
    INSERT INTO user_tasks (user_id{1}) VALUES ('{0}'{2})
  </env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

And then Bid Manager will transform the SQL command to:

```
INSERT INTO user_tasks (user_id, description, start_time, end_time, location) VALUES
('Willy', 'Meeting', '2006-1-16 10:00:00', '2006-1-16 12:00:00', 'Pittsburg')
```

Where {0} is replaced with “**willy**”, and {1} is replaced with *optionNameList*, which is **“, description, start_time, end_time, location”**, and {2} is replaced with *optionValueList*, which is **“, 'Meeting', '2006-1-16**

10:00:00', '2006-1-16 12:00:00', 'Pittsburg'".

Let's see the last sample, SetUser (MsgID: 2303):

Key Vector: {"MsgID", "UserId", "UserPwd", "TgtUserId", "role.opt", "is_expert.opt", "rate.opt"}

Value Vector: {"2600.1", "admin", "1234", "Willy", "3", "1", "20"}

The corresponding SQL command setting is:

```
<env-entry>
  <env-entry-name>setUser</env-entry-name>
  <!-- 0:user_id 1:optionUpdateList -->
  <env-entry-value>
    UPDATE user_info SET user_id = '{0}'{1} WHERE user_id = '{0}'
  </env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

And then Bid Manager will transform the SQL command to:

```
UPDATE user_info SET user_id = 'Willy', role = '3', is_expert = '1', rate = '20' WHERE user_id = 'Willy'
```

Where {0} is replaced with "willy", and {1} is replaced with *optionUpdateList*, which is "**, role = '3', is_expert = '1', rate = '20'**".

You may modify the entire SQL commands configuration in *web.xml* to match your database schema. But there is one thing you must keep in mind: Using ">" or "<", but not ">" or "<".

Below is the entire sample SQL commands configuration in *web.xml*.

```
<env-entry>
  <env-entry-name>user_login</env-entry-name>
  <!-- 0:user_id, 1:passwd 2:optionConditionList -->
  <env-entry-value>SELECT * FROM user_info WHERE disabled = 0 AND user_id = '{0}' AND passwd = '{1}'{2}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>list_bidroom</env-entry-name>
  <!-- 0:optionConditionList -->
  <env-entry-value>SELECT bidroom_id BidRoomId, bidroom_name BidRoomName, bidroom_descr BidRoomDescr, (SELECT COUNT(*) FROM bid_entity WHERE bidding_rooms.bidroom_id = bid_entity.bidroom_id AND canceled = 0 AND status = 0) BidNum FROM bidding_rooms WHERE deleted = 0{0}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>listBidRoomBids</env-entry-name>
  <!-- 0:bidroom_id 1:optionConditionList -->
  <env-entry-value>SELECT * FROM bid_entity WHERE canceled = 0 AND bidroom_id = {0}{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addNewBid.checkBid</env-entry-name>
  <!-- 0:bidroom_id 1:optionConditionList -->
  <env-entry-value>SELECT COUNT(*) FROM bid_entity WHERE canceled = 0 AND status = 0 AND bidroom_id = {0}{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addNewBid.addBid</env-entry-name>
  <!-- 0:user_id, 1:bidroom_id, 2:start_time, 3:end_time 4:optionNameList 5:optionValueList -->
  <env-entry-value>INSERT INTO bid_entity (starter_id, bidroom_id, start_time, end_time{4}) VALUES ('{0}', '{1}', '{2}', '{3}'{5})</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addNewBid.getBidId</env-entry-name>
  <!-- None -->
  <env-entry-value>SELECT @@IDENTITY</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>placeBid.checkIfClosed</env-entry-name>
  <!-- 0:bid_id, 1:bidroom_id -->
```

```

    <env-entry-value>SELECT status FROM bid_entity WHERE canceled = 0 AND bid_id = {0} AND
bidroom_id = {1}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>placeBid.checkIfExist</env-entry-name>
    <!-- 0:bid_id, 1:bidroom_id, 2:user_id -->
    <env-entry-value>SELECT COUNT(*) FROM bidding_status WHERE retracted = 0 AND bid_id =
{0} AND bidroom_id = {1} AND user_id = '{2}'</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>placeBid.checkSchedule</env-entry-name>
    <!-- 0:bid_id, 1:bidroom_id, 2:user_id -->
    <env-entry-value>SELECT user_tasks.description Description, user_tasks.location
Location, user_tasks.start_time Start_time, user_tasks.end_time End_time FROM bid_entity,
user_tasks WHERE bid_entity.canceled = 0 AND user_tasks.user_id = '{2}' AND
bid_entity.bid_id = {0} AND bid_entity.bidroom_id = {1} AND ((user_tasks.start_time <
bid_entity.task_start) AND (bid_entity.task_start < user_tasks.end_time)) OR
((user_tasks.start_time < bid_entity.task_end) AND (bid_entity.task_end <
user_tasks.end_time))</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>placeBid.placeBid</env-entry-name>
    <!-- 0:bid_id, 1:bidroom_id, 2:user_id 3:optionNameList 4:optionValueList -->
    <env-entry-value>INSERT INTO bidding_status (bid_id, bidroom_id, user_id, place_date,
Qob{3}) SELECT {0}, {1}, '{2}', GETDATE(), DATEPART(millisecond, GETDATE()){4} FROM
bid_entity, user_info WHERE bid_entity.bid_id = {0} AND bid_entity.bidroom_id = {1} AND
user_info.user_id = '{2}'</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>updateBidderTaskAndInfo.updateTask</env-entry-name>
    <!-- 0:user_id, 1:bid_id, 2:bidroom_id, 3:start_time, 4:end_time -->
    <env-entry-value>INSERT INTO user_tasks (user_id, description, start_time, end_time,
location) SELECT '{0}', bid_descr, '{3}', '{4}', location FROM bid_entity WHERE bid_id =
{1} AND bidroom_id = {2}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>updateBidderTaskAndInfo.updateUserInfo</env-entry-name>
    <!-- 0:user_id -->
    <env-entry-value>UPDATE user_info SET num_bids_won = num_bids_won + 1 WHERE user_id =
'{0}'</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>getBidders</env-entry-name>
    <!-- 0:bid_id -->
    <env-entry-value>SELECT * FROM bidding_status WHERE retracted = 0 AND result = 0 AND
bid_id = {0} ORDER BY Qob DESC</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>getBiddersByQob</env-entry-name>
    <!-- 0:bid_id 1:Qob -->
    <env-entry-value>SELECT * FROM bidding_status WHERE retracted = 0 AND result = 0 AND
bid_id = {0} AND Qob = {1}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>listClosingBids</env-entry-name>
    <!-- None -->
    <env-entry-value>SELECT * FROM bid_entity WHERE end_time < GETDATE() AND canceled
= 0 AND status = 0</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>listUserOngoingBids</env-entry-name>
    <!-- 0:user_id -->
    <env-entry-value>SELECT * FROM bid_entity WHERE canceled = 0 AND status = 0 AND user_id
= '{0}'</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>listBidders</env-entry-name>
    <!-- 0:bid_id 1:optionConditionList -->
    <env-entry-value>SELECT * FROM bidding_status WHERE bid_id = {0}{1}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>queryBid</env-entry-name>
    <!-- 0:bid_id 1:optionConditionList -->
    <env-entry-value>SELECT * FROM bid_entity WHERE bid_id = {0}{1}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>

```

```

<env-entry>
  <env-entry-name>getHighestQoB</env-entry-name>
  <!-- 0:bid_id -->
  <env-entry-value>SELECT MAX(Qob) FROM bidding_status WHERE retracted = 0 AND result =
0 AND bid_id = {0}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>getHoursSumByQoB</env-entry-name>
  <!-- 0:bid_id 1:Qob -->
  <env-entry-value>SELECT SUM(place_time) FROM bidding_status WHERE retracted = 0 AND
result = 0 AND bid_id = {0} AND Qob = {1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>listUserBids</env-entry-name>
  <!-- 0:user_id 1:optionConditionList -->
  <env-entry-value>SELECT bidding_status.*, bid_entity.bid_descr FROM bidding_status,
bid_entity WHERE bidding_status.retracted = 0 AND bid_entity.canceled = 0 AND
bidding_status.bid_id = bid_entity.bid_id AND bidding_status.user_id =
'{0}'{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>listFeasibleBids</env-entry-name>
  <!-- 0:user_id 1:optionConditionList -->
  <env-entry-value>SELECT bid_entity.* FROM bid_entity, user_tasks WHERE
bid_entity.canceled = 0 AND bid_entity.status = 0 AND user_tasks.user_id = '{0}' AND NOT
(((user_tasks.start_time &lt; bid_entity.task_start) AND (bid_entity.task_start &lt;
user_tasks.end_time)) OR ((user_tasks.start_time &lt; bid_entity.task_end) AND
(bid_entity.task_end &lt; user_tasks.end_time))) {1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>listStartedBids</env-entry-name>
  <!-- 0:user_id 1:optionConditionList -->
  <env-entry-value>SELECT * FROM bid_entity WHERE canceled = 0 AND starter_id =
'{0}'{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addTask</env-entry-name>
  <!-- 0:user_id 1:optionNameList 2:optionValueList -->
  <env-entry-value>INSERT INTO user_tasks (user_id{1}) VALUES
('{0}'{2})</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>deleteTask</env-entry-name>
  <!-- 0:user_id 1:task_id -->
  <env-entry-value>DELETE FROM user_tasks WHERE user_id = '{0}' AND task_id =
{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>updateTask</env-entry-name>
  <!-- 0:user_id 1:task_id 2:optionValueList -->
  <env-entry-value>UPDATE user_tasks SET user_id = '{0}'{2} WHERE user_id = '{0}' AND
task_id = {1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>listTasks</env-entry-name>
  <!-- 0:user_id 1:optionConditionList -->
  <env-entry-value>SELECT * FROM user_tasks WHERE user_id = '{0}'{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>checkTaskCollision</env-entry-name>
  <!-- 0:user_id 1:ignoreTaskId 2:startTime 3:endTime -->
  <env-entry-value>SELECT COUNT(*) FROM user_tasks WHERE user_id = '{0}' AND ((( '{2}' &lt;=
start_time) AND (start_time &lt; '{3}')) OR (( '{2}' &lt;= end_time) AND (end_time &lt;=
'{3}')) AND NOT task_id = {1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>cancelBid</env-entry-name>
  <!-- 0:starter_id, 1:bid_id, 2:bidroom_id 3:reason 4:optionValueList -->
  <env-entry-value>UPDATE bid_entity SET canceled = 1, cancel_reason = '{3}'{4} WHERE
canceled = 0 AND starter_id = '{0}' AND bid_id = {1} AND bidroom_id = {2}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>retractBid</env-entry-name>
  <!-- 0:user_id, 1:bid_id, 2:bidRoom_id 3:reason 4:optionValueList -->
  <env-entry-value>UPDATE bidding_status SET retracted = 1, retract_reason = '{3}'{4}
WHERE retracted = 0 AND user_id = '{0}' AND bid_id = {1} AND bidroom_id = {2}</env-entry-value>

```

```

    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>getRolePermission</env-entry-name>
  <!-- 0:user_id 1:passwd 2:msg -->
  <env-entry-value>SELECT role_permission.permission FROM role_permission JOIN user_info
ON user_info.user_id = '{0}' AND user_info.passwd = '{1}' AND user_info.role =
role_permission.role WHERE role_permission.msg = {2}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addUser.checkUserId</env-entry-name>
  <!-- 0:user_id -->
  <env-entry-value>SELECT COUNT(*) FROM user_info WHERE user_id = '{0}'</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addUser.addUser</env-entry-name>
  <!-- 0:user_id 1:optionNameList 2:optionValueList -->
  <env-entry-value>INSERT INTO user_info (user_id{1}) VALUES
('{0}'{2})</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>delUser</env-entry-name>
  <!-- 0:user_id 1:optionUpdateList -->
  <env-entry-value>UPDATE user_info SET disabled = 1{1} WHERE user_id =
'{0}'</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>getUser</env-entry-name>
  <!-- 0:user_id 1:optionConditionList -->
  <env-entry-value>SELECT * FROM user_info WHERE user_id = '{0}'{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>setUser</env-entry-name>
  <!-- 0:user_id 1:optionUpdateList -->
  <env-entry-value>UPDATE user_info SET user_id = '{0}'{1} WHERE user_id =
'{0}'</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addBidroom.checkBidRoomName</env-entry-name>
  <!-- 0:bidroom_name 1:optionConditionList -->
  <env-entry-value>SELECT COUNT(*) FROM bidding_rooms WHERE deleted = 0 AND bidroom_name
= '{0}'{1}</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addBidroom.addBidroom</env-entry-name>
  <!-- 0:bidroom_name 1:optionNameList 2:optionValueList -->
  <env-entry-value>INSERT INTO bidding_rooms (bidroom_name{1}) VALUES
('{0}'{2})</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addBidroom.getBidRoomId</env-entry-name>
  <!-- None -->
  <env-entry-value>SELECT @@IDENTITY</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>delBidroom</env-entry-name>
  <!-- 0:bidroom_id 1:optionUpdateList -->
  <env-entry-value>UPDATE bidding_rooms SET deleted = 1{1} WHERE bidroom_id =
'{0}'</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>starterUpdateBid.checkBid</env-entry-name>
  <!-- 0:starter_id 1:bidroom_id 2:bid_id -->
  <env-entry-value>SELECT status FROM bid_entity WHERE starter_id = '{0}' AND bidroom_id
= {1} AND bid_id = {2} AND canceled = 0</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>starterUpdateBid.updateBid</env-entry-name>
  <!-- 0:starter_id 1:bidroom_id 2:bid_id 3:optionUpdateList -->
  <env-entry-value>UPDATE bid_entity SET starter_id = '{0}', bidroom_id = {1}{3} WHERE
starter_id = '{0}' AND bidroom_id = {1} AND bid_id = {2} AND canceled = 0</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
  <env-entry-name>addCategory.checkCategory</env-entry-name>
  <!-- 0:bidroom_id 1:cat_parent 2:cat_name 3:cat_link -->

```

```

    <env-entry-value>SELECT c1.cat_id AS PARENT, (c1.id_path + '/' + CAST(c1.cat_id AS
VARCHAR)) AS IDPATH, (c1.name_path + '/' + c1.cat_name) AS NAMEPATH, c2.cat_id AS LINK,
c3.cat_id AS CHILD FROM category AS c1 LEFT JOIN category AS c2 ON c1.bidroom_id =
c2.bidroom_id AND c2.disabled = 0 AND c2.cat_link = 0 AND c2.cat_id = {3} LEFT JOIN category
AS c3 ON c1.bidroom_id = c3.bidroom_id AND c1.cat_id = c3.cat_parent AND c3.disabled =
0 AND c3.cat_name = '{2}' WHERE c1.bidroom_id = {0} AND c1.is_leaf = 0 AND c1.cat_id = {1}
UNION SELECT 0, '', '', c2.cat_id, c3.cat_id FROM category AS c1 LEFT JOIN category AS c2
ON c1.bidroom_id = c2.bidroom_id AND c2.disabled = 0 AND c2.cat_link = 0 AND c2.cat_id =
{3} LEFT JOIN category AS c3 ON c1.bidroom_id = c3.bidroom_id AND c3.cat_parent = 0 AND
c3.disabled = 0 AND c3.cat_name = '{2}' WHERE c1.bidroom_id = {0} AND {1} =
0</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>addCategory.addCategory</env-entry-name>
    <!-- 0:bidroom_id 1:cat_parent 2:cat_name 3:cat_link 4:id_path 5:name_path
6:optionNameList 7:optionValueList -->
    <env-entry-value>INSERT INTO category (bidroom_id, cat_name, cat_parent, cat_link,
id_path, name_path{6}) VALUES ({0}, '{2}', {1}, {3}, '{4}', '{5}'{7})</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>addCategory.getCategoryId</env-entry-name>
    <!-- None -->
    <env-entry-value>SELECT @@IDENTITY</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>delCategory</env-entry-name>
    <!-- 0:bidroom_id 1:cat_id 2:optionUpdateList -->
    <env-entry-value>UPDATE category SET disabled = 1{2} WHERE bidroom_id = {0} AND cat_id
= {1}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>listSubCategory</env-entry-name>
    <!-- 0:bidroom_id 1:cat_id 2:optionConditionList -->
    <env-entry-value>SELECT c0.cat_id, c0.cat_name, c0.name_path, (SELECT COUNT(*) FROM
category AS c1 LEFT JOIN category AS c2 ON c1.bidroom_id = c2.bidroom_id AND c2.disabled
= 0 AND (c1.cat_id = c2.cat_id OR c2.name_path LIKE (c1.name_path + '/' + c1.cat_name +
'%')) JOIN bid_entity AS b ON c2.bidroom_id = b.bidroom_id AND c2.cat_id = b.category AND
b.canceled = 0{2} WHERE c1.disabled = 0 AND c1.cat_id = c0.cat_id) AS BidNum FROM category
AS c0 WHERE c0.cat_link = 0 AND c0.disabled = 0 AND c0.bidroom_id = {0} AND c0.cat_parent
= {1} UNION SELECT c0.cat_link, c0.cat_name + '@', c0.name_path, (SELECT COUNT(*) FROM
category AS c1 LEFT JOIN category AS c2 ON c1.bidroom_id = c2.bidroom_id AND c2.disabled
= 0 AND (c1.cat_id = c2.cat_id OR c2.name_path LIKE (c1.name_path + '/' + c1.cat_name +
'%')) JOIN bid_entity AS b ON c2.bidroom_id = b.bidroom_id AND c2.cat_id = b.category AND
b.canceled = 0{2} WHERE c1.disabled = 0 AND c1.cat_id = c0.cat_link) AS BidNum FROM category
AS c0 WHERE c0.cat_link > 0 AND c0.disabled = 0 AND c0.bidroom_id = {0} AND c0.cat_parent
= {1}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>getCategory</env-entry-name>
    <!-- 0:bidroom_id 1:cat_id -->
    <env-entry-value>SELECT * FROM category WHERE bidroom_id = {0} AND cat_id = {1} AND
cat_link = 0 AND disabled = 0 UNION SELECT c2.* FROM category c1 JOIN category c2 ON c1.cat_link
= c2.cat_id AND c1.bidroom_id = c2.bidroom_id AND c2.disabled = 0 WHERE c1.bidroom_id =
{0} AND c1.cat_id = {1} AND c1.cat_link > 0 AND c1.disabled = 0</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>getBid</env-entry-name>
    <!-- 0:bidroom_id 1:bid_id 2:optionConditionList -->
    <env-entry-value>SELECT * FROM bid_entity WHERE bidroom_id = {0} AND bid_id =
{1}{2}</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>micropayment.getAccount</env-entry-name>
    <!-- 0:user_id -->
    <env-entry-value>SELECT pointvalue, bonusvalue FROM member WHERE id =
'{0}'</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>
    <env-entry-name>micropayment.log</env-entry-name>
    <!-- 0:id, 1:typeid, 2:mercid, 3:termid, 4:cdate, 5:ctime, 6:lidm, 7:amount, 8:bonus,
9:memo -->
    <env-entry-value>INSERT INTO PointsLog (id, typeid, mercid, termid, cdate, ctime, lidm,
amount, bonus, memo) VALUES ('{0}', {1}, '{2}', '{3}', '{4}', '{5}', '{6}', {7}, {8},
'{9}')</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
</env-entry>

```

Implement your closing bid class

It is impossible to write a wizard class which can handle any kind of closing bid and update the related columns' value. So we allow you to write your own code to handle the closing bid and then add your implementation into the Bid Manager. The mechanism works as follow:

- 0: Write a class implements the “*tw.org.itri.icl.chronobot.bid.ClosingBidHandle*” interface. This interface has three method:

```
public boolean isSupport(ResultSetMetaData rsmd);  
  
public void setBidList(ResultSet rs);  
  
public void closeBid();
```

In your class constructor, you may register your own class by calling:

```
tw.org.itri.icl.chronobot.bid.ClosingBidHandleManager.registerBidHandle((tw.org  
.itri.icl.chronobot.bid.ClosingBidHandle)this);
```

Then your class can dynamic register to *ClosingBidHandleManager* by calling:

```
Class.forName("YourClassName").newInstance();
```

- 1: Add to *BidHandleClassList* in *ChronoServerWebService/WEB-INF/web.xml*.

```
<env-entry>  
  
  <env-entry-name>BidHandleClassList</env-entry-name>  
  
  <env-entry-value>  
  
    tw.org.itri.icl.chronobot.bid.ChronobotBidHandle;xx.yy.zz.YourClassName  
  
  </env-entry-value>  
  
  <env-entry-type>java.lang.String</env-entry-type>  
  
</env-entry>
```

BidHandleClassList is a semicolon-separate list, so you should separate different class name with a semicolon character, ";".

- 2: You can using *ClosingBidHandleManager.getClosingBidHandle(ResultSet rs)* method to get a class which can handle a closing bid *ResultSet*. This method will call each registered class's *isSupport(ResultSetMetaData rsmd)* method to determine if this class support that *ResultSet*.
- 3: In your class's *isSupport(ResultSetMetaData rsmd)* method, you should check if the *ResultSetMetaData* has every column name you will reference to or update the value. If so, then return *true*, otherwise return *false*.

Chronobot JSP Tag Library

In JavaServer Pages technology, *actions* are elements that can create and access programming language objects and affect the output stream. Since JSP v1.1, JSP supports the development of reusable modules called *custom actions*. A custom action is invoked by using a *custom tag* in a JSP page. A *tag library* is a collection of *custom tags*. Before the availability of custom actions, JavaBeans components in conjunction with scriptlets were the main mechanism for performing processing, accessing databases and other enterprise services such as email and directories, and flow control. The disadvantage of using this approach is that it makes JSP pages more complex and difficult to maintain. JSP tag libraries are created by developers who are proficient at the Java programming language and expert in accessing data and other services. JSP tag libraries are used by Web application designers who can focus on presentation issues rather than being concerned with how to access databases and other enterprise services. That's why we provide Chronobot Tag Library. We encapsulate most of the Chronobot Web Service messages so that you can ignore the entire Chronobot protocols, data encoding, and message definitions. All Chronobot tags that have a *var* attribute also have a *scope* attribute that can be used to indicate the scope of the JSP variable: **page**, **request**, **session**, or **application**. Below is the Chronobot action tags lists and descriptions:

List Tag Result Interface

The Result interface is used to retrieve information from objects returned from all <Chronobot:List*> tags. We use the *javax.servlet.jsp.jstl.sql.Result* interface as the Result. For complete information about this interface, see the API documentation for the [JSTL](#) packages.

```
public interface Result
{
    public String[] getColumnNames();
    public int getRowCount();
    public Map[] getRows();
    public Object[][] getRowsByIndex();
    public boolean isLimitedByMaxRows();
}
```

The *var* attribute set by a *List* tag is of type *Result*. The *getRows* method returns an array of maps that can be supplied to the *items* attribute of a *forEach* tag. The JSTL expression language converts the syntax `${result.rows}` to a call to *result.getRows*. The expression `${BidRoomList.rows}` in the following example returns an array of maps.

When you provide an array of maps to the *forEach* tag, the *var* attribute set by the tag is of type Map. To retrieve information from a row, use the *get("colname")* method to get a column value. The JSP expression language converts the syntax `${map.colname}` to a call to `map.get("colname")`. For example, the expression `${BidRoom.bidRoomId}` returns the value of the *bidRoomId* entry of a *BidRoom* map.

```
<c:forEach var="BidRoom" varStatus="row"
items="${BidRoomList.rows}">
  <tr>
    <td>${BidRoom.bidRoomId}</td>
    <td>${BidRoom.bidRoomName}</td>
    <td>${BidRoom.bidRoomDescr}</td>
  </tr>
</c:forEach>
```

<chronobot:Attribute>

For parameterized Chronobot request, we provide a nested *Attribute* tag inside most Chronobot tags. *Attribute* tag will add request option parameter to the outer tag. In this example, *Attribute* tag add “status=0” and “category=6” request option parameters to ListBidRoomBid message.

```
<chronobot:ListBidRoomBid var="bidList" userId="${usrID}"
userPwd="${usrPwd}" bidRoomId="6" startRow="${param.startRow}"
maxRows="${maxRows}">
  <chronobot:Attribute name="status" value="0"/>
  <chronobot:Attribute name="category" value="6"/>
</chronobot:ListBidRoomBid>
```

<chronobot:ListBidRoom>

User asks BM to list all the active bid rooms. BM returns the list of active bid rooms.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | Page |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```

<chronobot:ListBidRoom var="BidRoomList" userId="\${usrID}"
userPwd="\${usrPwd}"/>
<c:forEach var="BidRoom" varStatus="row"
items="\${BidRoomList.rows}">
  <tr>
    <td align="center">\${BidRoom.bidRoomId}</td>
    <td align="center">\${BidRoom.bidRoomName}</td>
    <td>\${BidRoom.bidRoomDescr}</td>
  </tr>
</c:forEach>

```

<chronobot:ListBidRoomBid>

User requests BM to list bids in the bid room.

Attributes

| Name | Description | EL | Type | Required | Default |
|-----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | Page |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room | Yes | int | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```

<chronobot:ListBidRoomBid var="activeBidList"
userId="\${sessionScope.usrID}" userPwd="\${sessionScope.usrPwd}"
bidRoomId="\${BidRoom.bidRoomId}" startRow="\${param.startRow}"
maxRows="\${maxRows}"/>

```

<chronobot:ListPlacedBid>

User requests BM to list the bids s/he has been placed.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | Page |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```
<chronobot:ListPlacedBid var="placedBidList "  
userId="$ {sessionScope.usrID} " userPwd="$ {sessionScope.usrPwd} "  
startRow="$ {param.placedBidStartRow} " maxRows="$ {maxRows} " />
```

<chronobot:ListFeasibleBid>

The user requests TKM to list all feasible bids according to his schedule (time) and capability (knowledge).

Attributes

| Name | Description | EL | Type | Required | Default |
|----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | Page |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```
<chronobot:ListFeasibleBid var="bidList "  
userId="$ {sessionScope.usrID} " userPwd="$ {sessionScope.usrPwd} " />
```

<chronobot:ListStartedBid>

User requests BM to list the bids s/he started.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | Page |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```
<chronobot:ListStartedBid var="startedBidList "  
userId="$ {sessionScope.usrID} " userPwd="$ {sessionScope.usrPwd} " />
```

<chronobot:ListTask>

User requests TKM to return all information of the given task in his/her time schedule.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | Page |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```
<chronobot:ListTask var="taskList" userId="${sessionScope.usrID}"
userPwd="${sessionScope.usrPwd}"/>
```

<chronobot:StartNewBid>

User requests BM to start a new bid on his/her behalf. Properties of the bid are given below. BM returns if the request was successful. In case of a failure, reason for the failure (of start of the bid) must be specified in response message. On Success, the optional properties of the response message describe the result of the active bid. Moreover, BM returns a unique id for this bid (it's unique per bid room). Any messages from the user pertaining to this bid should carry the bid id and bid room id along with other parameters.

Attributes

| Name | Description | EL | Type | Required | Default |
|-------------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room | Yes | int | Yes | None |
| bidDuration | The duration of the bid | Yes | String | No | NULL |
| endDateTime | The end time of the bid | Yes | String | No | NULL |

Sample

```
<chronobot:StartNewBid var="newBid" userId="${sessionScope.usrID}"
userPwd="${sessionScope.usrPwd}" bidRoomId="${param.BidRoomId}"
bidDuration="${param.BidDuration} Minutes">
<chronobot:Attribute name="bid_title" value="${param.Title}"/>
    <chronobot:Attribute name="category" value="${param.cat_id}"/>
    <chronobot:Attribute name="bid_descr" value="${param.descr}"/>
    <chronobot:Attribute name="max_price" value="200"/>
    <chronobot:Attribute name="multi_win" value="1"/>
    <chronobot:Attribute name="keywords" value="${param.Title}"/>
```

```

</chronobot:StartNewBid>
<h4>${newBid.Success ? 'Succeeded' : 'Failed'}</h4>
<ul type="square">
  <c:if test="${newBid.Success}">
    <li>Bid ID:${newBid.BidId}</li>
  </c:if>
  <li>Bid Title:${param.Title}</li>
  <li>Bid Description:${param.Description}</li>
  <li>Create Time:'${newBid.start_time}'</li>
  <c:if test="${not newBid.Success}">
    <li><strong>Reason:${newBid.Reason}</strong></li>
  </c:if>
</ul>

```

<chronobot:PlaceBid>

User tells BM that s/he wants to place a bid for the given bid id. BM responds to the bidder if the bid was successful. Failure reasons are returned if request failed.

Attributes

| Name | Description | EL | Type | Required | Default |
|---------------|---|-----|---------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room | Yes | int | Yes | None |
| bidId | Id of the bid | Yes | int | Yes | None |
| allowMultiBid | Allow or not one user places the same bid more than once. | Yes | boolean | No | NULL |

Sample

```

<chronobot:PlaceBid var="placeBid" userId="${sessionScope.usrID}"
userPwd="${sessionScope.usrPwd}" bidRoomId="${param.BidRoomId}"
bidId="${param.BidId}">
  <chronobot:Attribute name="place_time"
value="${param.BidPlacedTime}" />
</chronobot:PlaceBid>

```

<chronobot:AddUser>

User requests BM to accept a new registration. BM returns whether the registration was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|-----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| newUserId | Target user id to be added. | Yes | String | Yes | None |

Sample

```
<chronobot:AddUser var="result" userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" newUserId="{param.NewUserId}">
  <chronobot:Attribute name="name" value="{param.NewUser}" />
</chronobot:AddUser>
```

<chronobot:DelUser>

User requests BM to delete the given user. BM returns whether the deletion was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|--------------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| deleteUserId | Target user id to be deleted. | Yes | String | Yes | None |

Sample

```
<chronobot:DelUser var="result" userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" deleteUserId="{param.deleteUserId}" />
```

<chronobot:GetUser>

User requests BM to show user profile of the given user id. BM returns complete user information.

Attributes

| Name | Description | EL | Type | Required | Default |
|--------------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| targetUserId | Id of target user whose profile is to be retrieved. | Yes | String | Yes | None |

Sample

```
<chronobot:GetUser var="user" userId="{sessionScope.usrID}"  
userPwd="{sessionScope.usrPwd}" targetUserId="{param.targetUserId}"/>
```

<chronobot:SetUser>

User requests BM to update a profile. Users can update their personal data and administrators may change profiles of other people. BM returns the updated profile of the given user, just as the GetUser message does.

Attributes

| Name | Description | EL | Type | Required | Default |
|--------------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| targetUserId | Id of target user whose profile is to be updated. | Yes | String | Yes | None |

Sample

```
<chronobot:SetUser var="user" userId="{sessionScope.usrID}"  
userPwd="{sessionScope.usrPwd}" targetUserId="{param.targetUserId}">  
  <chronobot:Attribute name="name" value="{param.NewUser}"/>  
</chronobot:SetUser>
```

<chronobot:AddBidRoom>

User (administrator) requests BM to create a bid room. BM will allocate a unique id for this room and return the id to user if everything was successful. Failure reasons of creation are also returned.

Attributes

| Name | Description | EL | Type | Required | Default |
|-------------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomName | Name of the new bid room. Has to be unique else the request will be rejected. | Yes | String | Yes | None |

Sample

```
<chronobot:AddBidRoom userId="admin" userPwd="admin"  
bidRoomName="{param.bidRoomName}">
```

```

    <chronobot:Attribute name="bidroom_descr"
value="{param.bidRoomDescr}"/>
</chronobot:AddBidRoom>

```

<chronobot:DelBidRoom>

User (administrator) requests BM to delete a bid room. The failure can be either the bid room id does not exist or there are active bids in the bid room.

Attributes

| Name | Description | EL | Type | Required | Default |
|-------------|---|-----------|-------------|-----------------|----------------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room that is to be deleted. | Yes | int | Yes | None |

Sample

```

<chronobot:DelBidRoom userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" bidRoomId="{param.bidRoomId}"/>

```

<chronobot:StarterUpdateBid>

Bid starter requests BM to update parameter values of his/her bid. Changeable parameters of an active bid are listed as request option parameters.

Attributes

| Name | Description | EL | Type | Required | Default |
|-------------|---|-----------|-------------|-----------------|----------------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room that is to be updated. | Yes | int | Yes | None |
| bidId | Id of the bid. | Yes | int | Yes | None |

Sample

```

<chronobot:StarterUpdateBid userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" bidRoomId="{param.bidRoomId}"
bidId="{param.bidId}">
    <chronobot:Attribute name="bid_title" value="{param.Title}"/>
</chronobot:StarterUpdateBid>

```

<chronobot:AddCategory>

User (administrator) requests BM to add a new category in the bid room.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------------|--|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room in which to add the category. | Yes | int | Yes | None |
| parentCatId | Where this category is located. | Yes | int | Yes | None |
| categoryName | The category's name. | Yes | String | Yes | None |
| linkCategoryId | The linked category's id, if any. | Yes | int | No | 0 |

Sample

```
<chronobot:StarterUpdateBid userId="${sessionScope.usrID}"
userPwd="${sessionScope.usrPwd}" bidRoomId="${param.bidRoomId}"
bidId="${param.bidId}">
  <chronobot:Attribute name="bid_title" value="${param.Title}"/>
</chronobot:StarterUpdateBid>
```

<chronobot:AddCategory>

User (administrator) requests BM to add a new category in the bid room.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------------|--|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room in which to add the category. | Yes | int | Yes | None |
| parentCatId | Where this category is located. | Yes | int | Yes | None |
| categoryName | The category's name. | Yes | String | Yes | None |
| linkCategoryId | The linked category's id, if any. | Yes | int | No | 0 |

Sample

```
<chronobot:AddCategory userId="${sessionScope.usrID}"
userPwd="${sessionScope.usrPwd}" bidRoomId="${param.bidRoomId}"
parentCatId="${param.categoryId}"
```

```

categoryName=" ${param.newCategoryName} "
linkCategoryId=" ${param.linkTo} ">
  <chronobot:Attribute name="is_leaf" value=" ${param.isLeaf == 'yes' ?
'1' : '0'} "/>
</chronobot:AddCategory>

```

<chronobot:DelCategory>

User (administrator) requests BM to del a category in the bid room.

Attributes

| Name | Description | EL | Type | Required | Default |
|------------|--|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room in which to del the category. | Yes | int | Yes | None |
| categoryId | Id of the category. | Yes | int | Yes | None |

Sample

```

<chronobot:DelCategory userId=" ${sessionScope.usrID} "
userPwd=" ${sessionScope.usrPwd} " bidRoomId=" ${param.bidRoomId} "
categoryId=" ${param.categoryId} "/>

```

<chronobot>ListCategory>

User asks BM to list all the sub-category in current bid room and current category.
BM returns the list of categories.

Attributes

| Name | Description | EL | Type | Required | Default |
|------------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room in which to list the category. | Yes | int | Yes | None |
| categoryId | Id of the current category. | Yes | int | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```

<chronobot>ListCategory var="categoryList"

```

```

userId="${sessionScope.usrID}" userPwd="${sessionScope.usrPwd}"
bidRoomId="${param.BidRoomId}" categoryId="${param.categoryId}">
    <chronobot:Attribute name="b.status" value="0"/>
</chronobot:ListCategory>

```

<chronobot:GetCategory>

User asks BM to show category of the given category id. BM returns complete category information.

Attributes

| Name | Description | EL | Type | Required | Default |
|------------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room in which to list the category. | Yes | int | Yes | None |
| categoryId | Id of the current category. | Yes | int | Yes | None |

Sample

```

<chronobot:GetCategory var="currentCategory" userId="${usrID}"
userPwd="${usrPwd}" bidRoomId="${param.BidRoomId}"
categoryId="${param.categoryId}"/>

```

<chronobot:QueryBid>

User requests BM to get a bid's open information.

Attributes

| Name | Description | EL | Type | Required | Default |
|---------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | "Page" |
| userId | User's ID | Yes | String | Yes | None |
| userPwd | User's password | Yes | String | Yes | None |
| bidId | Id of the bid to be queried. | Yes | int | Yes | None |

Sample

```

<chronobot:QueryBid var="bid" userId="${usrID}" userPwd="${usrPwd}"
bidId="${param.BidId}"/>

```

<chronobot:ListBidder>

User requests BM to get a bid's bidders.

Attributes

| Name | Description | EL | Type | Required | Default |
|----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidId | Id of the bid to be queried. | Yes | int | Yes | None |
| startRow | The start row of the result | Yes | int | No | 1 |
| maxRows | The max row of the result | Yes | int | No | -1(all) |

Sample

```
<chronobot:ListBidder var="bidderList" userId="{usrID}"
userPwd="{usrPwd}" bidId="{param.BidId}"/>
```

<chronobot:CancelBid>

The administrator tells BM s/he wants to cancel the bid. BM returns whether the canceling was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|-----------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| starterId | User id of the person starting the bid. | Yes | String | Yes | None |
| bidRoomId | Id of the bid room which contains the bid to be canceled. | Yes | int | Yes | None |
| bidId | Id of bid to be canceled. | Yes | int | Yes | None |
| reason | Why admin cancel the bid. | Yes | String | No | EMPTY |

Sample

```
<chronobot:CancelBid var="result" userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" starterId="{param.DefaultUser}"
bidRoomId="{param.BidRoomId}" bidId="{param.BidId}"/>
```

<chronobot:StarterCancelBid>

The bid starter tells BM s/he wants to cancel the bid. BM returns whether the canceling was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|------|-------------------------|----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |

| | | | | | |
|-----------|---|-----|--------|-----|--------|
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room which contains the bid to be canceled. | Yes | int | Yes | None |
| bidId | Id of bid to be canceled. | Yes | int | Yes | None |
| reason | Why the starter cancel the bid. | Yes | String | No | EMPTY |

Sample

```
<chronobot:StarterCancelBid var="result"
userId="{sessionScope.usrID}" userPwd="{sessionScope.usrPwd}"
bidRoomId="{param.BidRoomId}" bidId="{param.BidId}" />
```

<chronobot:RetractBid>

The administrator tells BM that s/he want to retract the bid. BM returns whether the retracting was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|-----------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidderId | User id of the person bidding the bid. | Yes | String | Yes | None |
| bidRoomId | Id of the bid room which contains the target bid. | Yes | int | Yes | None |
| bidId | Bid id on which the person is withdrawing. | Yes | int | Yes | None |
| reason | Why admin retract the bid. | Yes | String | No | EMPTY |

Sample

```
<chronobot:RetractBid var="result" userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" bidderId="{param.DefaultUser}"
bidRoomId="{param.BidRoomId}" bidId="{param.BidId}" />
```

<chronobot:BidderRetractBid>

The bidder tells BM that s/he want to retract the bid. BM returns whether the retracting was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|------|-------------|----|------|----------|---------|
|------|-------------|----|------|----------|---------|

| | | | | | |
|-----------|---|-----|--------|-----|--------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room which contains the bid to be canceled. | Yes | int | Yes | None |
| bidId | Bid id on which the person is withdrawing. | Yes | int | Yes | None |
| reason | Why the bidder retract the bid. | Yes | String | No | EMPTY |

Sample

```
<chronobot:BidderRetractBid var="result"
userId="{sessionScope.usrID}" userPwd="{sessionScope.usrPwd}"
bidRoomId="{param.BidRoomId}" bidId="{param.BidId}" />
```

<chronobot:SignIn>

User requests BM to accept his/her entry of bid system.

Attributes

| Name | Description | EL | Type | Required | Default |
|---------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |

Sample

```
<chronobot:SignIn var="checkID" userId="{param.account}"
userPwd="{param.password}" />
```

<chronobot:PlaceThenCloseBid>

User tells BM that s/he wants to place a bid for the given bid id and then close it. BM responds to the bidder if the request was successful. Failure reasons are returned if request failed.

Attributes

| Name | Description | EL | Type | Required | Default |
|-----------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room | Yes | int | Yes | None |
| bidId | Id of the bid | Yes | int | Yes | None |

| | | | | | |
|---------------|---|-----|---------|----|------|
| allowMultiBid | Allow or not one user places the same bid more than once. | Yes | boolean | No | NULL |
|---------------|---|-----|---------|----|------|

Sample

```
<chronobot:PlaceThenCloseBid var="placeBid"
userId="{sessionScope.usrID}" userPwd="{sessionScope.usrPwd}"
bidRoomId="1" bidId="{param.BidId}">
    <chronobot:Attribute name="place_time" value="{param.num_hrs}"/>
</chronobot:PlaceThenCloseBid>
```

<chronobot:CloseBid>

The administrator tells BM s/he wants to close the bid. BM returns whether the request was successful.

Attributes

| Name | Description | EL | Type | Required | Default |
|-----------|---|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| bidRoomId | Id of the bid room which contains the bid to be canceled. | Yes | int | Yes | None |
| bidId | Id of bid to be canceled. | Yes | int | Yes | None |

Sample

```
<chronobot:CloseBid var="result" userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" bidRoomId="{param.BidRoomId}"
bidId="{param.BidId}"/>
```

<chronobot:MicroPayment>

Call MicroPayment System to pay or withdraw amount of value. The amount value is the sum of point and bonus. If not zero, use bonus first, then pay remain amount by point.

Attributes

| Name | Description | EL | Type | Required | Default |
|---------|----------------------------------|-----|--------|----------|---------|
| var | The name of the result. | No | String | Yes | None |
| scope | The scope of <i>var</i> variable | No | String | No | “Page” |
| userId | User’s ID | Yes | String | Yes | None |
| userPwd | User’s password | Yes | String | Yes | None |
| mercId | Merchant ID. | Yes | String | Yes | None |
| termId | Terminal ID | Yes | String | Yes | None |

| | | | | | |
|--------|--------------------------|-----|-----|----|---|
| amount | Total amount of payment. | Yes | int | No | 0 |
| point | The point to pay | Yes | int | No | 0 |
| bonus | The bonus to use | Yes | int | No | 0 |

Sample

```
<chronobot:MicroPayment var="payment" userId="{sessionScope.usrID}"
userPwd="{sessionScope.usrPwd}" mercId="0000000001"
termId="00000001" bonus="{param.price}"/>
```