Classic Chat API – Alpha v3

Chat Channels are a defining aspect of the legacy Battle.net experience. Modernization efforts have or will limit functions the community has created over the years. That's not acceptable, so Classic Games is excited to debut our official Chat API.

Alpha 3 of CAPI is now available for testing on production, with your help and support we will expand the feature set over time.

API Key

An API key is required to connect. The key is unique to your bot and should not shared or checked into source control.

Connect to Battle.net using StarCraft Remastered, Diablo 2 or Warcraft 3 and once in your preferred channel (Op or Clan) use the command **/register-bot** to start the registration process. An email is required to be registered to the account and after executing the command an email with activation link will be sent to the email on file. The user is required to click on the link to get a valid API Key. Executing the command a second time will issue a new API Key and invalidate the old key.

Connection Endpoint

The Chat API uses JSON with UTF8 encoding as its protocol with secure websockets as the transport. The connection endpoint will be:

wss://connect-bot.classic.blizzard.com/v1/rpc/chat

It is recommended that the certificate is checked to ensure that the common name matches *.classic.blizzard.com

The bot can send requests to the server and will get responses back and the server will send events to the bot as well. All requests/responses will have these main keys;

- command - lets you know what type of payload is attached
- payload - body of request
- request id - allows tracking of request/responses
- status - reporting of errors (see below)

Area	Code	Reason
8	1	Not Connected to chat
8	2	Bad request
6	5	Request timed out
6	8	Hit rate limit

API: Authentication

When connection is established, the client will need to send an authentication request with the API key:

```
Request:
{
  "command": "Botapiauth.AuthenticateRequest",
  "request_id": 1,
  "payload": {
    "api_key": "[API KEY]"
  }
}

Response:
{
  "command": "Botapiauth.AuthenticateResponse",
  "request_id": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: ChatConnect

Connect the bot to the gateway and chat channel

```
Request:
{
  "command": " Botapichat.ConnectRequest",
  "request_id": 1,
  "payload": {
  }
}

Response:
{
  "command": "Botapichat.ConnectResponse",
  "request_id": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}

Async Event:
{
  "command": "Botapichat.ConnectEventRequest",
  "request_id": 1,
  "payload": {
    "channel": "Op Lodle"
  }
}
```

API: ChatDisconnect

Disconnects the bot from the gateway and chat channel

```
Request:
{
  "command": " Botapichat.DisconnectRequest",
  "request_id": 1,
  "payload": {
  }
}

Response:
{
  "command": "Botapichat.DisconnectResponse",
  "request_id": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}

Async Event:
{
  "command": "Botapichat.DisconnectEventRequest", "request_id":
  1,
  "payload": {
  }
}
```

API: ChatSendMessage

Sends a chat message to the channel

```
Request:
{
  "command": "Botapichat.SendMessageRequest",
  "requestId": 1,
  "payload": {
     "message": "[MESSAGE]"
  }
}

Response:
{
  "command": "Botapichat.SendMessageResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: ChatSendWhisper

Sends a chat message to one user in the channel

```
Request:
{
  "command": "Botapichat.SendWhisperRequest",
  "requestId": 1,
  "payload": {
     "message": "[MESSAGE]",
     "user_id": "[USER ID]"
  }
}

Response:
{
  "command": "Botapichat.SendWhisperResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: ChatBanUser

Bans a user from the channel

```
Request:
{
  "command": "Botapichat. BanUserRequest",
  "requestId": 1,
  "payload": {
     "user_id": "[USER ID]"
  }
}

Response:
{
  "command": "Botapichat.BanUserResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: ChatUnbanUser

Un-Bans a user from the channel

```
Request:
{
  "command": "Botapichat.UnbanUserRequest",
  "requestId": 1,
  "payload": {
     "toon_name": "[TOON NAME]"
  }
}

Response:
{
  "command": "Botapichat.UnbanUserResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
```

```
}
```

API: ChatSendEmote

Sends an emote on behalf of a bot

```
Request:
{
  "command": "Botapichat.SendEmoteRequest",
  "requestId": 1,
  "payload": {
     "message": "[EMOTE MESSAGE]"
  }
}

Response:
{
  "command": "Botapichat.SendEmoteResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: ChatKickUser

Kicks a user from the channel

```
Request:
{
  "command": "Botapichat.KickUserRequest",
  "requestId": 1,
  "payload": {
     "user_id": "[USER ID]"
  }
}

Response:
{
  "command": "Botapichat.KickUserResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: ChatSetModerator

Sets the current chat moderator to a member of the current chat. Same as a normal user doing **/designate** followed by **/resign**

```
Request:
{
  "command": "Botapichat.SendSetModeratorRequest",
  "requestId": 1,
  "payload": {
     "user_id": "[USER ID]"
  }
}

Response:
{
  "command": "Botapichat. SendSetModeratorResponse",
  "requestId": 1,
  "status": {
    "area": 0,
    "code": 0,
  },
  "payload": {
  }
}
```

API: OnMessageEvent

A message was posted to the channel

```
{
  "command": "Botapichat.MessageEventRequest",
 "payload": {
    "user_id": "[USER ID]",
    "message": "[MESSAGE]",
    "type": "[TYPE]",
  }
}
```

Type is one of: `Whisper, Channel, ServerInfo, ServerError, Emote`

API: OnUserUpdateEvent

A user has joined the current channel or got an update

```
{
  "command": "Botapichat.UserUpdateEventRequest",
 "payload": {
    "user_id": "[USER ID]",
    "toon_name": "[TOON NAME]",
    "flags": [
        "[FLAG 1]",
        "[FLAG 2]"
    ],
    "attributes": {
       "[key]": "[value]",
    }
  }
}
```

Flags are: Admin, Moderator, Speaker, MuteGlobal, MuteWhisper Attributes are: ProgramId, Rate, Rank, Wins

API: OnUserLeaveEvent

A user in the current channel has left

```
{
  "command": "Botapichat.UserLeaveEventRequest",
 "payload": {
```

```
    "user_id": "[USER ID]"
  }
}
```