Classic Chat API – Alpha v2

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 1 of CAPI is now available for testing on the PTR. This first iteration allows for interactive bots. 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.

Required Information:

- Blizzard account in good standing
- Legacy Gateway - limit of one gateway per bot
- Legacy Account Name - bot will have the same name as the account with a bot prefix on the end
- Legacy Account Email - must match existing account
- Legacy Channel

Register Here:

https://goo.gl/forms/39jk4jPQVb8X4Aqa2

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: 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: 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]"
   }
}
```