

Protocol for WEB API for Members

NCMS (CD Segment)

Version 1.0



The NSE Clearing Limited (National Clearing)
Exchange Plaza, Plot No. C/1, G Block,
Bandra-Kurla Complex, Bandra (E),
Mumbai - 400 051

NSE Clearing Confidential

Notice

© Copyright NSE Clearing Ltd (NCL). All rights reserved. Unpublished rights reserved under applicable copyright and trades secret laws.

The contents, ideas and concepts presented herein are proprietary and confidential.
Duplication and disclosure to others in whole, or in part is prohibited

Revision History

Date	Change Description	Edited By	Version
13-Apr-2026	Initial version		1.0

Table of Contents

Revision History.....	2
Introduction.....	4
<i>General Instructions</i>	4
<i>HTTP Status Codes</i>	4
<i>Common Error Response JSON</i>	5
<i>Environment Details</i>	5
<i>API Consumer Registration</i>	5
<i>API Security</i>	6
Clearing Corporation APIs	7
<i>POST /<version>/request/token</i>	7
<i>POST /<version>/inquire/trd-act</i>	9
<i>POST /<version>/request/approval-rejection</i>	15
<i>POST /<version>/request/cp-modification</i>	18
<i>POST /<version>/request/approve-all</i>	21
Appendix A - Reference Codes.....	24
<i>Market Type</i>	24
<i>Market Status</i>	24
<i>Transaction Code</i>	24
<i>Activity Type</i>	24
<i>Book Type</i>	25
<i>Client Type</i>	25
<i>Buy Sell Flag</i>	25
<i>Trade Status</i>	25
<i>Option Type</i>	26
<i>Is Approval Flag</i>	26
<i>Action Type</i>	26
<i>Exchange Code</i>	27
Appendix B - Response Codes.....	27
<i>HTTP response code</i>	27
<i>Message based response code</i>	28

Introduction

This document provides information on the Web APIs used for programmatic access to trade management related data between NCL's NCMS Platform and its Members. It details the messaging protocols and structures required to develop this interface.

General Instructions

1. Following headers need to be provided in all API calls made to clearing corporation.
 - **Content-Type:** This header should be provided in all requests with method as "POST". Header value should be "application/json".
 - **User-Agent:** All requests should contain this header. The value of "User-Agent" header can be "/".
 - **Accept-Encoding:** This header is required in all API calls to CC. The value of this header should be blank.
 - **Accept:** This header value should be "application/json"
2. Some of the key specifications related to JSON and standards followed for the API's are as follows
 - JSON is built on 2 structures. Map containing key value pairs and an ordered list of values.
 - A value could be boolean (true / false), number, decimal, String or a structure (List or Object).
 - Object or key value pair structure consists of keys which are strings and values of any of the above types. E.g. {"name":"Amit", "age":25}
 - List contains list of values. E.g. ["Amit", "Ajay", "Vikas"]
 - A Boolean has only 2 values true or false.
 - String values are enclosed in double quotes. e.g. "name", "Amit", "Pending"
 - Numbers and decimals are represented without any thousand - separator character. Decimal indicator is dot (".")
 - Numbers have an optional maximum number of digits. If not specified, then it is defaulted to 18.
 - Decimals have 2 mandatory length parameters. The first length parameter indicates number of digits in the whole part (before decimal place) and the second length parameter indicates number of digits in the decimal part (after decimal place).
3. All URLs for API will be always in lower case.
4. All JSON field names will follow camel-hump style of naming. A field with multiple words would be concatenated without spaces. All characters will be in lower case. First characters of words other than the first word in the field name will be in upper case. For e.g. field for "Order Number" could be represented by field name "orderNumber". Other examples are "firstName", "lastName".
5. In case of JSONs representing an object or a key-value pair, keys with null values could be omitted from the JSON.

HTTP Status Codes

All API's will respond with an HTTP status code. A status code of 200 would indicate successful execution of the API and the response body would be as defined in the API specification.

In case of an error a HTTP status code other than 200 will be returned. The API may or may not return an error response JSON depending upon the type of error encountered. Following are the HTTP status codes that could be returned by the APIs

#	Status Code	Reason	Description
1	200	SUCCESS	Request was handled successfully
2	400	BAD REQUEST	Indicates a validation / business logic error / json parsing errors
3	401	UNAUTHORIZED: Failed to authenticate the request	Indicates that the credentials / access token shared for authentication is invalid or expired.
4	404	NOT FOUND	Incorrect URL or Resource does not exist
5	405	METHOD_NOT_ALLOWED	Unsupported HTTP Method: A request was made for a resource using a request method not supported by that resource (e.g. using GET instead of POST).
6	500	UNKNOWN_ERROR	Internal Server Error. Such errors are to be reported to the support desk.
7	503	SVC_UNAVAILABLE	Service unavailable.

Common Error Response JSON

Field	Type	Mandatory	Description
code	Number	Yes	Http Status Code. See above
messages	List<String>	Yes	One or more error messages

Sample Response

```
{
  "code": 400,
  "messages": [
    "Invalid JSON."
  ]
}
```

Environment Details

Base URL for all CD Segment Trade Management API endpoints mentioned in this document will be as follows:

Testing Environment: <https://uat.connect2nsccl.com/CD/TRD/>

Live Environment: <https://www.connect2nsccl.com/CD/TRD/>

API Consumer Registration

To initiate data consumption through the API endpoints, members are required to submit necessary information, including their IP address and registered email address, to NCL. Additionally, members must provide their public key certificates to NCL to enable payload encryption. The public key should be generated using the RSA algorithm and comply with the X.509 standard to ensure compatibility. Once this information is received, the member will be registered for API access and provided with a Consumer Key and Secret.

API Security

OAuth 2.0, an industry-standard authorisation protocol, is employed to facilitate access to API endpoints. Members can generate bearer tokens through the designated API call (refer to details below). The token response payload's data field will be asymmetrically encrypted using the Member's Public Key Certificate with the RSA algorithm. This encrypted payload will be delivered as a Base64-encoded string.

Furthermore, an AES secret key and IV unique to the member will be included within the access token payload and retained by both NCL and the member. This will serve to enable secure encryption and decryption of API payloads.

Clearing Corporation APIs

This chapter gives details of the API's exposed by clearing corporation and to be consumed by members.

POST /<version>/request/token

To obtain a token, the member's consumer app must request for the access token using API POST /<version>/request/token endpoint. The access token can be reused to access NCL API data until it expires (after 'n' minutes). During API registration, the member receives a consumer key and secret, which are validated for token authorization. The access token payload also contains aes_secret_key and aes_iv required to decrypt response payloads.

Request

Get Token Request Header Parameters

#	Parameter Name	Data Type	Description	Sample Value
1	Authorization	String	The format should be as follows: Basic <member_credentials> Here, member_credentials refers to a base64-encoded string consisting of the following data: cons_key:cons_secret	Basic MRZmwzCl6.....SGq lCaxH9rAM3hVIMJzFg==
2	nonce	String	A nonce uniquely identifies each server request. It should be a base64-encoded string in the format: ddMMyyyyHHmmssSSS:<6-digit random number>.	MjAwMTlwMTcxNjEyMjE0TE6
3	grant_type	String	Value MUST be set to "client_credentials".	client_credentials

Sample Request

```
POST /request/token HTTP/1.1
Host: www.connect2nsccl.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic MRZmwzCl6.....SGqXlCaxH9rAM3hVIMJzFg==
nonce: MjAwMTlwMTcxNjEyMjE0TE6ODk0MjY3
x-www-form-urlencoded
grant_type=client_credentials
```

Response

The response's data field includes the encrypted token response payload as a Base64-encoded string. To access the raw token payload, first decode the Base64 data string, then decrypt the resulting bytes using the Member private key associated with the public certificate provided during the API Consumer Registration process.

Success Response Sample

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "data":
  "GHZovnrYUaw6X8J9GE1vfLLlgb6b/KIVp6B0uKttHP91FIFNEpEZIMI43eWMcyOUEsvqr5fj4snHA125K8++8U/R
tCYC7r3bW+2U/P6J/nG2qNtFGRoM1Koc0KVMcFgNptJC6BK2Bs6Fo44KAOtJ97NBIf9R0/WPxJy3dqi2A6zXo9t
qn22JfgaFq/2JWZT0kX1grGkBEJZZImUiA0+ftpV3JfqrnYwZAtCr+cM7nbhab8Mri8cWBeHNG1pALU/A1jcDvar5
/NTdMDSCImkuw7ngXQpnOFX1mP1AITAYLHOTnuau3KoE653lze2+ruleMuk9celEuL+vahYqtZfz7w=="
}
```

Failure Response Sample

```
HTTP/1.1 401 UNAUTHORIZED
Content-Type: application/json
```

```
{
  "messages":{"code":"0100401"},
  "status":"error"
}
```

Token Response Raw Parameters

#	Parameter Name	Data Type	Description	Sample Value
1	access_token	String	The access token that is issued by the authorization server.	ee1073de-45d0-4040-b9c2-eddfa80280c0
2	token_type	String	The type of the token issued.	bearer
3	expires_in	int	The lifetime in seconds of the access token. For example, the value "32400" denotes that the access token will expire in nine hour from the time the response was generated.	32400
4	Scope	String	If identical to the scope requested by the client otherwise, REQUIRED.	api_scope
5	key	String	aes_secret_key and aes_iv collectively used to encrypt and decrypt further API request-response	
6	iv	String	aes_secret_key and aes_iv collectively used to encrypt and decrypt further API request-response	

Sample output of the decrypted **raw token payload** in JSON format:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token": "ee1073de-45d0-4040-b9c2-eddfa80280c0",
  "token_type": "bearer",
  "expires_in": "3600",
  "scope": "api_scope",
  "key": "aes_secret_key",
  "iv": "aes_iv"
}
```


POST /<version>/inquire/trd-act

This API will allow members to inquire for trades & actions using API POST /<version>/inquire/trd-act endpoint. A maximum of 1,00,000 messages can be inquired in a single API call.

Request

Request Header Parameters

#	Parameter Name	Data Type	Description	Sample Value
1	Authorization	String	Bearer token encrypted using the AES received as part of token response. <access_token>	Basic MRZmwzdkje382jdw8ue93j dCaxH9rAM3hVIMJzFg==
2	nonce	String	A nonce uniquely identifies each server request. It should be a base64-encoded string in the format: ddMMyyyyHHmmssSSS:<6-digit random number>.	MjAwMTIwMTcxNjEyMjE0TE6
3	consumerKey	String	The Member Consumer Key received as part of API Registration process. <consKey>	consKey

Request Body Payload (JSON)

#	Parameter Name	Data Type	Description	Sample Value
1	Version	String	API version	1.0
2	Data.msgId	String	Unique request number for each request <CODE><YYYYMMDD><nnnnnnnn> Member / Custodian Code (Length: 4 or 5) <ul style="list-style-type: none"> YYYYMMDD – Date format nnnnnnnn – Running sequence no. starting from one i.e. For first request of the day, it should be (0000001). 	XXXXX201310140000001
3	data.dataformat	String	Request data format: Response data format	CSV:CSV
4	data.tradesInquiry	CSV	Data Structure specified below	0,ALL,TMTRADES,CPTRADES, ERRORACT,

Trade Action Inquiry (tradesInquiry) Request Packet Structure (CSV)

Field Name	Description	Data Type	Size (in Bytes)	Sample
seqNo	Trade sequence till where server had sent the trade	long	8	0

Field Name	Description	Data Type	Size (in Bytes)	Sample
	information in the previous request. For first download request of the day, it should be 0.			
srchFilter	Search Filter	String	50	<ul style="list-style-type: none"> ALL – All Trades TMTRADES – Clearing Member to view trades as a Trading Member (Only applicable to Clearing Members) CPTRADES – Clearing Member to view only CP Trades (Only applicable to Clearing Members)
fill1	Filler	String	10	Leave it blank
fill2	Filler	String	10	Leave it blank

Sample Request

Request Header:

```
POST /request inquire/trd-act HTTP/1.1
Host: uat.connect2nsccl.com
Authorization: Basic MRZmwzdkje382jdw8ue93jdCaxH9rAM3hVIMJzFg==
consumerKey: consKey
nonce: MjAwMTlwMTcxNjEyMjE1OTE6ODk0MjY3
Content-Type: application/json
```

Request Body:

```
{
  "data":
  "i3fJhLKZHhGdanX8csAP4sfqaXse/PO2ek84FMMocd8hLMVgHuOQREft6QsruHisVrqTBjDqAL4guyyVLLV3RNR
YRRa3uuhRj+BdJI7UJE.....A6dy/yJaem0qa40X+5iUvteGpQ7BlpQ=="
}
```

Sample output of the decrypted **request body payload data** in JSON format:

The access token in the Authorization header, as well as the data parameter in the request body, are required to be AES-encrypted. When making an API call, the Base64-encoded string of these encrypted values must be used. Members should perform encryption using the AES secret key and IV provided at the time of token generation alongside the access token.

```
{
  "version": "1.0",
  "data":
  {
    "msgId": "00001201310140000001",
    "dataFormat": "CSV:CSV",
    "tradesInquiry": "0, ALL,,"
  }
}
```

```
}
}
```

The access token in the authorization header, as well as the data parameter in the request body, are required to be AES-encrypted. When making an API call, the Base64-encoded string of these encrypted values must be transmitted. Members should perform encryption using the AES secret key and IV provided at the time of token generation alongside the access token.

Response

Response Payload Structure (JSON)

#	Parameter Name	Data Type	Description	Sample Value
1	status	String	Response Status	success/error
2	messages.code	String	Refer to section “Message based response code”	01010000
3	data.msgId	String	Unique request number sent back in the response	XXXXX201310140000001
4	data.tradeActionInquiry	CSV	Data Structure specified below. Structure Separator “^”	Refer Sample Response

Trade Action Inquiry (tradeActionInquiry) Response Packet Structure (CSV – Structure Separator “^”)

Field Name	Description	Data Type	Size (in Bytes)	Sample
sysinfoResData	Control Record Structure	CSV	-	System Info Response Structure given below
trdActResData	Detail Record Structure	CSV	-	Field Separator – “,” Record Separator – “^” Trades Response Data Structure given below

Control Record Structure (sysinfoResData) (CSV)

Field Name	Description	Data Type	Size (in Bytes)	Sample
mktSts	Market Status Refer Section “Transcodes”	Short	2	1
currTrdDate	Current Trade Date (YYYYMMDD)	Long	8	20251028
sfill1	Filler	String	10	
sfill1	Filler	String	10	
maxSeqNo	Max sequence number sent in response	long	8	2513977
noOfRec	Count of trades sent in the response	int	4	2

Detail Record Structure (trdActResData) (CSV) (Field Separator – “,” | Record Separator – “^”)

#	Field Name	Description	Data Type	Size (in Bytes)	Sample
1	seqNo	Unique Sequence Number	long	8	2014127
2	mktType	Market Type. Refer Section “Reference Codes”	String	1	1
3	trdNo	Trade Number	Double	8	91114327
4	trdTime	Trade Time in jiffy format	Long	8	87841606323214
5	trdSecToken	Securities Token	String	11	42401
6	Quantity	Trade Quantity	Int	4	200
7	trdPr	Trade price in paise	Float	4	1295
8	bsFlg	Buy/Sell Flag. Refer Section “Reference Codes - Buy Sell Flag”	String	1	2
9	ordNo	Order Number	Double	8	1100000000435542
10	trdTmBrnCd	TM Branch code	Short	2	3
11	usrId	User Id	int	4	31908
12	proCli	Client Type. Refer Section “Reference Codes”			2
13	trdAcc	Client Account No	String	20	XXXXX
14	cpCd	Custodial participant Id	String	12	CP0000000012
15	remarks	Remarks	String	25	
16	actType	Activity Type. Refer Section “Reference Codes”	Short	2	2
17	transCd	Transaction Code. Refer Section “Reference Codes”	Short	2	6001
18	ordTm	Order Time in milliseconds from 1980	long	8	1340356541
19	booktype	Book Type. Refer Section “Reference Codes”	Int	2	1
20	oppTmCd	Opposite Broker Id	String	1	
21	ctclId	CTCL code	double	8	110005129066000
22	status	Trade Status. Refer Section “Reference Codes”	String	1	P
23	trdTmCd	TM code	String	5	10001
24	trdSecSymbl	Securities Symbol	String	10	USDINR
25	trdSecSeries	Securities Series	String	2	
26	inst	Instrument	String	6	OPTCUR
27	expDt	Expiry Date (in milliseconds from 1980)	int	4	1340461800
28	strPrc	Strike Price in paise	int	4	90
29	optType	Option Type for Option Contract. Refer Section “Reference Codes”	String	2	PE
30	exchangeCd	Exchange Code. Refer Section “Reference Codes”	Short	2	1
31	trdUniqId	Trade Unique Id	String	20	42401911143271
32	errCd	Action Response Error Code	Short	2	0

#	Field Name	Description	Data Type	Size (in Bytes)	Sample
		Refer Section Async Response Codes			
33	actDTm	Action Date Time Date time in milli seconds from 1980	Int	4	1397898700
34	actID	Action Type Refection Section Reference Code	Short	2	1
35	trdTime	Trade time Date time in milli seconds from 1980	Int	4	1397898700
36	cmCD	Primary Member Code of the Clearing Member	String	5	XXXXX
37	Ccid	Clearing Corporation ID for Clearing Members of Trading member Refer section "CC ID"	Short	2	1
38	Msgid	Value of field data.msgId provided in request	String	20	00001201310140000001
39	Filler1	Filler	Double	8	
40	Filler2	Filler	Double	8	
41	Filler3	Filler	String	16	
42	Filler4	Filler	String	16	
43	Filler5	Filler	String	16	

Sample Failure Response

Wrong access token or expired access token

HTTP/1.1 401 UNAUTHORIZED

Content-Type: application/json

```
{
  "messages":{"code":"0100401"},
  "status":"error"
}
```

Error in encryption

HTTP/1.1 400 BAD_REQUEST

Content-Type: application/json

```
{  
  "messages":{"code":"0100400"},  
  "status":"error"  
}
```

Sample Success Response

The payload in the response to the API call, will be AES-encrypted string. The Base64-encoded string of this encrypted value will be transmitted. Members should perform decryption using the AES secret key and IV provided at the time of token generation alongside the access token.

Actual Response

HTTP/1.1 200 OK

Content-Type: application/json

```
{  
  "status": "success",  
  "messages":  
  {  
    "code": "01010000"  
  },  
  "data":  
  "i1iw0PPNS0DJNSX8bswCpY65aWYSobTpBCR/UZAqacBiO8smQeHRa338+Ro7qGi8VOXSVzPCEP04  
oXWHd/AjOozKTge2vO9WiOJdJY3VJVhdcwZL3Gj4tEbXi4vxft+SfJ9bRxyfh5kMHZXvclnC55mpkHca9  
fdgm8pKmT0SdnQsKMJ11GMUYxKQtDvdzQXyxWI4GY3f"  
}
```

Response with Raw Data

```
{  
  "status": "success",  
  "messages": {  
    "code": "01010000"
```

```

},
"data": {
{
"msgId": "00001201310140000001",
"tradeActionInquiry": "1,20251028,,2513977,2^2014127,1,91114327,878441606323214,42401,200,1295,2,
1100000000435542,3,31908,2,XXXXX,CP00000000012,,2,6001,1340356541,1,,110005129066000,P,10001,US
DINR,,OPTCUR,1340461800,90,PE,1,42401911143271,0,1397898700,1,1397898700,XXXXX,1,0000120131
0140000001,,,,,^2015123,1,91114344,978441606321000,52402,100,1295,2,1100000000435542,3,31908,
2,XXXXX,CP00000000011,,2,6001,1340356541,1,,110005129066000,P,10001,USDINR,,OPTCUR,1340461800,
90 ,PE,1,42401911143271,0,1397898700,1,1397898700,XXXXX,1,00001201310140000001,,,,,"
}
}

```

POST /<version>/request/approval-rejection

Request

This API will allow clearing members for approval and rejection of modification requests

Request Header Parameters

#	Parameter Name	Data Type	Description	Sample Value
1	Authorization	String	Bearer token encrypted using the AES received as part of token response. <access_token>	Basic MRZmwzdkje382jdw8ue93j dCaxH9rAM3hVIMJzFg==
2	nonce	String	A nonce uniquely identifies each server request. It should be a base64-encoded string in the format: ddMMyyyyHHmmssSSS:<6-digit random number>.	MjAwMTIwMTcxNjEyMjE0 TE6
3	consumerKey	String	The Member Consumer Key received as part of API Registration process. <consKey>	consKey

Request Body Payload (JSON)

#	Parameter Name	Data Type	Description	Sample Value
1	Version	String	API version	1.0
2	Data.msgId	String	Unique request number for each request <CODE><YYYYMMDD><nnnnnnnn>	XXXXX201310140000001

#	Parameter Name	Data Type	Description	Sample Value
			Member Code (Length: 5) • YYYYMMDD – Date format • nnnnnnn – Running sequence no. starting from one i.e. For first request of the day, it should be (0000001).	
3	data.isApproval	Char	Approval request or Rejection Request	Y – Approval N – Rejection
4	data.appRejData	JSON	Array of Approval or Rejection structure	Max 50000 records allowed per messageID. Structure details given below

Request Packet Structure:

#	Field Name	Description	Data Type	Size (in Bytes)	Sample
1	seqNo	Sequence number of trade	long	8	1111111
2	trdNo	Trade Number	long	8	1000
3	bsFlg	Buy/Sell Flag. Refer Section "Reference Codes - Buy Sell Flag"	Short	2	1
4	uniqId	Trade Unique ID	String	20	1544221684

Sample Request

Request Header:

```
POST /request/approval-rejection HTTP/1.1
Host: uat.connect2nsccl.com
Authorization: Basic MRZmwzdkje382jdw8ue93jdCaxH9rAM3hVIMJzFg==
consumerKey: consKey
nonce: MjAwMTIwMTcxNjEyMjE1OTE6ODk0MjY3
Content-Type: application/json
```

Request Body:

```
{
  "data":
  "i3fJhLKZHhGdanX8csAP4sfqaXse/PO2ek84FMMocd8hLMVgHuOQREft6QsruHisVrqTBjDqAL4guyyVLLV3RnrYRRa3uuhRj+BdJI7UJE.....A6dy/yJaem0qa40X+5iUvteGpQ7BlpQ=="
}
```

Sample output of the decrypted **request body payload data** in JSON format:

The access token in the Authorization header, as well as the data parameter in the request body, are required to be AES-encrypted. When making an API call, the Base64-encoded string of these encrypted values must be used. Members should perform encryption using the AES secret key and IV provided at the time of token generation alongside the access token.

```
{
  "version": "1.0",
  "data": {
    "msgId": "00240201310140000001",
    "isApproval": "Y",
    "appRejData": [
      {
        "seqNo": 1111111,
        "trdNo": 1000,
        "bsFlag": 1,
        "uniqId": "1544221684"
      },
      {
        "seqNo": 2222222,
        "trdNo": 2000,
        "bsFlag": 2,
        "uniqId": "22222222222222"
      }
    ]
  }
}
```

Response

Response Data Payload (JSON)				
Sr. No.	Parameter Name	Data Type	Description	Sample Value
1	status	String	Response status	success/error
2	messages.success	String	Message	Request submitted successfully
3	data.code	String	Refer Section “Message based response code”.	01010000

```
{
  "status": "Success",
  "messages": {
    "success": " Request submitted successfully."
  },
}
```

```

"data": {
  "code": "01010000"
}
}

```

POST /<version>/request/cp-modification

This API will allow members for CP modification using API POST /<version>/ request/cp-modification

Below is the list of modifications which can be carried out through the CP modification API

- CP Trade to CP Trade Modification
- CP Trade to Client Modification
- Client to CP Trade Modification

Request

Request Header Parameters

#	Parameter Name	Data Type	Description	Sample Value
1	Authorization	String	Bearer token encrypted using the AES received as part of token response. <access_token>	Basic MRZmwzdkje382jdw8ue93j dCaxH9rAM3hVIMJzFg==
2	nonce	String	A nonce uniquely identifies each server request. It should be a base64-encoded string in the format: ddMMyyyyHHmmssSSS:<6-digit random number>.	MjAwMTIwMTcxNjEyMjE0 TE6
3	consumerKey	String	The Member Consumer Key received as part of API Registration process. <consKey>	consKey

Request Body Payload (JSON)

#	Parameter Name	Data Type	Description	Sample Value
1	Version	String	API version	1.0
2	Data.msgId	String	Unique request number for each request <CODE><YYYYMMDD><nnnnnnnn> Member Code (Length: 5) • YYYYMMDD – Date format • nnnnnnnn – Running sequence no. starting from one i.e. For first	XXXXX201310140000001

#	Parameter Name	Data Type	Description	Sample Value
			request of the day, it should be (0000001).	
3	data. cpModData	JSON	Array of CP Modification structure	Max 50000 records allowed per message ID. Structure details given below

Request Packet Structure:

#	Field Name	Description	Data Type	Size (in Bytes)	Sample
1	seqNo	Sequence number of trade	long	8	1111111
2	ordNo	Order No	Long	12	2600000001529680
3	trdNo	Trade Number	long	8	2025042980000022
4	bsFlg	Buy/Sell Flag. Refer Section "Reference Codes - Buy Sell Flag"	Short	2	1
5	newCPCode	Below are the modifications which can be carried out and 1. CP Trade to CP Trade Modification (newCPCode: CP0000000012) 2. CP Trade to Client Modification (newCPCode: Blank) 3. Client to CP Trade Modification (newCPCode: CPDUMMY1111)	String	12	CP0000000012
6	uniqId	Trade Unique ID	String	15	111111111111111

Sample Request

Request Header:

```
POST /request/cp-modification HTTP/1.1
Host: uat.connect2nsccl.com
Authorization: Basic MRZmwzdkje382jdw8ue93jdCaxH9rAM3hVIMJzFg==
consumerKey: consKey
nonce: MjAwMTIwMTcxNjEyMjE1OTE6ODk0MjY3
Content-Type: application/json
```

Request Body:

```
{
  "data":
  "i3fJhLKZHhGdanX8csAP4sfqaXse/PO2ek84FMMocd8hLMVgHuOQREft6QsruHisVrqTBJDqAL4guyyVLLV3RNR
YRRa3uuhRj+BdJI7UJE.....A6dy/yJaem0qa40X+5iUvteGpQ7BlpQ=="
}
```

Sample output of the decrypted **request body payload data** in JSON format:

The access token in the Authorization header, as well as the data parameter in the request body, are required to be AES-encrypted. When making an API call, the Base64-encoded string of these encrypted values must be used. Members should perform encryption using the AES secret key and IV provided at the time of token generation alongside the access token.

➔ Sample request call (cp trade to cp trade modification)

```
{
  "version": "1.0",
  "data": {
    "msgId": "00240201310140000001",
    "dataFormat": "",
    "cpModData": [
      {
        "seqNo": "1111111 ",
        "ordNo": "2600000001529680",
        "trdNo": "2025042980000022",
        "bsFlg": 1,
        "newCPCode": " CP0000000012",
        "uniqlId": "111111111111111"
      }
    ]
  }
}
```

➔ Sample request call (cp trade to client modification)

```
{
  "version": "1.0",
  "data": {
    "msgId": "00240201310140000001",
    "dataFormat": "",
    "cpModData": [
      {
        "seqNo": "2222222",
        "ordNo": "2600000001526780",
        "trdNo": "2025042980000025",
        "bsFlg": 1,
        "newCPCode": "",
        "uniqlId": "222222222222222"
      }
    ]
  }
}
```

```
}

```

➔ Sample request call (Client to CP Trade Modification)

```
{
  "version": "1.0",
  "data": {
    "msgId": "00240201310140000001",
    "dataFormat": "",
    "cpModData": [
      {
        "seqNo": "3333333",
        "ordNo": "2600000001529999",
        "trdNo": "2025042980000026",
        "bsFlg": 1,
        "newCPCode": "CPDUMMY1111",
        "uniqlId": "33333333333333"
      }
    ]
  }
}
```

Response

Response Data Payload (JSON)				
Sr. No.	Parameter Name	Data Type	Description	Sample Value
1	status	String	Response status	success/error
2	messages.success	String	Message	Request submitted successfully
3	data.code	String	Refer Section “Message based response code”.	01010000

```
{
  "status": "Success",
  "messages": {
    "success": " Request submitted successfully."
  },
  "data": {
    "code": "01010000"
  }
}
```

POST /<version>/request/approve-all

This API will allow clearing members for to perform Approve All action API POST /<version>/request/approve-all

Request

Request Header Parameters

#	Parameter Name	Data Type	Description	Sample Value
1	Authorization	String	Bearer token encrypted using the AES received as part of token response. <access_token>	Basic MRZmwzdkje382jdw8ue93jdCaxH9rAM3hVIMJzFg==
2	nonce	String	A nonce uniquely identifies each server request. It should be a base64-encoded string in the format: ddMMyyyyHHmmssSSS:<6-digit random number>.	MjAwMTlwMTcxNjEyMjE1OTE6TE6
3	consumerKey	String	The Member Consumer Key received as part of API Registration process. <consKey>	consKey

Request Body Payload (JSON)

#	Parameter Name	Data Type	Description	Sample Value
1	Version	String	API version	1.0
2	Data.msgId	String	Unique request number for each request <CODE><YYYYMMDD><nnnnnnnn> Member Code (Length: 5) • YYYYMMDD – Date format • nnnnnnnn – Running sequence no. starting from one i.e. For first request of the day, it should be (0000001).	XXXXX201310140000001
3	Data.memCode	String	Member Code (Length: 5)	XXXXX

Sample Request

Request Header:

```
POST /request/approve-all HTTP/1.1
Host: uat.connect2nscl.com
Authorization: Basic MRZmwzdkje382jdw8ue93jdCaxH9rAM3hVIMJzFg==
consumerKey: consKey
nonce: MjAwMTlwMTcxNjEyMjE1OTE6ODk0MjY3
Content-Type: application/json
```

Request Body:

```
{
```

```

{
  "data":
    "i3fJhLKZHhGdanX8csAP4sfqaXse/PO2ek84FMMocd8hLMVgHuOQREft6QsruHisVrqTBjDqAL4guyyVLLV3RnrYRRa3uuhrJ+BdJI7UJE.....A6dy/yJaem0qa40X+5iUvteGpQ7BlpQ=="
}

```

Sample output of the decrypted **request body payload data** in JSON format:

The access token in the Authorization header, as well as the data parameter in the request body, are required to be AES-encrypted. When making an API call, the Base64-encoded string of these encrypted values must be used. Members should perform encryption using the AES secret key and IV provided at the time of token generation alongside the access token.

```

{
  "version": "1.0",
  "data":
    {
      "msgId": "00001201310140000001",
      "memCode": "00001",
    }
}

```

Response

Response Data Payload (JSON)				
Sr. No.	Parameter Name	Data Type	Description	Sample Value
1	status	String	Response status	success/error
2	messages.success	String	Message	Request submitted successfully
3	data.code	String	Refer Section "Message based response code".	01010000

```

{
  "status": "Success",
  "messages": {
    "success": " Request submitted successfully."
  },
  "data": {
    "code": "01010000"
  }
}

```

```
}
```

Appendix A - Reference Codes

Market Type

1	Normal
2	Odd Lot
3	Spot
4	Auction
5	Call Auction 1
6	Call Auction 2

Market Status

1	Preopen shutdown
2	Normal Market Preopen ended
3	Open Msg
4	Close Msg
5	Closing Start
6	Closing End

Transaction Code

6001	Original Trade
5525	Trade Modification Approval
5565	Control Trade Modification
5520	Trade Cancellation Approval
5560	Control Trade Cancellation
5530	Trade Cancellation Rejection
5445	Trade modification (Client Modification)
5440	Trade Cancellation

Activity Type

2	Original Trade
7	Trade Cancellation
101	Buy Participant modification
102	Sell Participant modification

103	Buy & Sell Participant modification
104	Quantity modification
105	Buy Account No. modification
106	Sell Account No. modification
107	Buy & Sell Account No.modification
109	Buy Trade Cancellation due to modification
110	Sell Participant Cancellation due to modification
111	Buy & Sell Trade Cancellation due to modification

Book Type

1	Regular Lot
2	Special Terms
3	Stop Loss / MIT
4	Negotiated Trade
5	Odd Lot
6	Spot
7	Auction
11	Call Auction 1
12	Call Auction 2

Client Type

1	Cli
2	Pro

Buy Sell Flag

1	BUY
2	SELL

Trade Status

P	Pending
R	Reject
A	Approve

Option Type

CA	Call American
PA	Put American
CE	Call European
PE	Put European

Is Approval Flag

Y	Approve
N	Reject

Action Type

2	Buy SI Generated
3	Sell SI Generated
4	AppRej Buy Approval
5	AppRej Sell Approval
6	Buy Side CP Modification (Old CP)
7	Sell Side CP Modification (Old CP)
8	Buy Side CP Modification (New CP)
9	Sell Side CP Modification (New CP)
14	AppRej Buy Rejected
15	AppRej Sell Rejected
16	Buy SI Cancelled
17	Sell SI Cancelled
20	Approval/Rejection Window is closed
22	Approve All window is closed
23	Buy CM is not matching
24	Sell CM is not matching
25	AppRej action is already done

26	Approve All action is already done
27	Approve All process window is open
28	AppRej record is locked
29	Approve All process is already running

Exchange Code

1	NSE Trades
2	BSE Trades
3	MSE Trades

Appendix B - Response Codes

There can be two types of response codes

- HTTP response codes
- Message based response codes
- Async response codes

HTTP response code

- HTTP responses shall be generated during login with success or failure status
- HTTP response shall also be generated in case of any authentication/input validation failure of the message
- HTTP response codes are as follows:

HTTP Response Codes			
Sr. No.	Reason	Meaning	HTTP Response Code
1	SUCCESS	Request was handled successfully	200
2	UNKNOWN_ERROR	Internal Server Error: Internal server error has occurred in our platform	500
3	SVC_UNAVAILABLE	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server	503
4	METHOD_NOT_ALLOWED	Unsupported HTTP method: A request was made for a resource using a request method not supported by that resource (e.g. using POST instead of GET)	405

HTTP Response Codes			
Sr. No.	Reason	Meaning	HTTP Response Code
5	BAD REQUEST	PARAMETER_ABSENT – There's a required parameter which is not present in the request	400
6	BAD REQUEST	DATA_INVALID – The data is not in correct format and not recognized by our system	400
7	BAD REQUEST	DATA_FORMAT_REJECTED – Unsupported Data format parameter value	400
8	UNAUTHORIZED: Failed to authenticate the request	CONSUMER_KEY_UNKNOWN – The provided Consumer Key (API key) is not registered in our system OR service is not registered	401
9	UNAUTHORIZED: Failed to authenticate the request	TOKEN_INVALID – The provided token is not registered in our system	401
10	UNAUTHORIZED: Failed to authenticate the request	UNAUTHORIZED: <ul style="list-style-type: none"> Unauthorized requestor IP address API access disabled 	401
11	PERMISSION_DENIED	Subscriber has temporarily disallowed access to his private data	403
12	The requested URL was not found	The requested URL was not found	404
13	REQUEST_NOT_FOUND	Registered request not found	570

Message based response code

- Message based response code shall be populated in the field “**code**” of the JSON response message
- It shall be of below format
 - First four characters (Field Identifier): refers to specific field or the entire message
 - Next characters (Validation code): refers to specific validation failure or success. Success code shall be populated only on successful acceptance of the message.

Field Identifier is as follows:

Sr. No.	Module	Field Name	Field Identifier
1	Entire Message	NA	0101
2	Input Data Parameter	msgId	0102
3	Input Data Parameter	dataFormat	0103
4	Input Data Parameter	msgPrepDt	0105
5	Input Data Parameter	msgPrepTm	0106
6	Input Data Parameter	isApproval	0109
7	Input Data Parameter	seqNo	0107

Sr. No.	Module	Field Name	Field Identifier
8	Input Data Parameter	srchFilter	0108
9	Input Data Parameter	noOfRec	0110
10	Input Data Parameter	appRejData	0111
11	Input Data Parameter	trdNo	0112
12	Input Data Parameter	bsFlag	0113
13	Input Data Parameter	uniqId	0114

Validation codes are as follows:

Sr. No.	Validation	Validation Type	Validation Code	Validation performed on Field
1	Submitted to server successfully	Message Level	0000	Entire Message
2	Duplicate request received	Message Level	0001	Entire Message
3	All HTTP status codes	HTTP error codes	HTTP Response codes. Refer section "HTTP Response Code".	Entire Message
4	Mismatch in control and data record	Message Level	0200	Entire Message
5	Minimum Required Length	Generic	0201	msgId
6	Maximum Required Length	Generic	0202	msgId
7	Mandatory field	Generic	0204	msgId, noOfRec, seqNo, srchFilter, trdDate, appRejData, trdNo, seqNo, uniqId, bsFlag
8	Data Format like Message Id / Date Format	Generic	0206	msgId, trdDate
9	Minimum allowed value	Generic	0207	seqNo, noOfRec
10	Maximum allowed value	Generic	0208	noOfRec
11	Invalid Value	Generic	0209	seqNo, isApproval, srchFilter, trdDate, bsFlag
12	System Error	Generic	0241	NA
13	Service Unavailable	Generic	0242	NA
14	Request Parsing Error: Invalid Request Structure	Generic	0243	NA
15	Invalid Member Code	Generic	0114	Msgid,
16	Invalid Date	Generic	0246	Msgid,
17	Window Closed	Generic	0122	NA

Sample example for success or failure code

- Example for Generic Error Code

Let's assume that msgId field holds value ABCD201340402132165, which turns out to be an error "Invalid Data Format". Error Code that will be generated is as shown below:

Field Identifier: 0102

Validation Code: 0206

code = combination of "Field Identifier" and "Validation Code" = 01020206

- Example for Field Error Code

Let's assume that seqNo field holds value -1, which turns out to be an error "Minimum allowed value". Error Code that will be generated is as shown below:

Field Identifier: 0107

Validation Code: 0207

code = combination of "Field Identifier" and "Validation Code" = 01070207

- Example for Success code (Submitted to server successfully)

Let's assume that message for approval/rejection is successful, success code that will be generated is as shown below:

Field Identifier: 0101 (which is the identifier of the entire message)

Validation Code: 0000

code = combination of "Field Identifier" and "Validation Code" = 01010000

- Example for HTTP error code

Let's assume that the invalid request scenario due to BAD Request, error code that will be generated is as shown below:

Field Identifier: 0101 (which is the identifier of the entire message)

Validation Code: 400

code = combination of "Field Identifier" and "Validation Code" = 0101400

Async response code

Async response code shall be populated in the field “errCd” of the message

Error	Error Code
Success	0
Trade in file should be available in System	1
Invalid Contract	2
Invalid Participant or TM code should be Active	3
TM code should match with that in system	4
Trade already cancelled	5
System Error	6
Trade already approved	7
Trade already rejected	8
Outstanding alert	9
Invalid user	10
Invalid data	11
Old CP should be in system	12
Clearing Member is Disabled. Trade Approval/Rejection not allowed.	13
Buy/Sell flag in trade should match that in system	15
Old CP and New CP should not be same	16
CP code less than or equal to 12 characters	17
Approve All request rejected-Invalid market status	19
Approval/Rejection Window is Closed	20
CP Modification Window is Closed	21
Invalid CP code	22
Buy CM is not Matching	23
Sell CM is not Matching	24
Already approved / rejected	25

Invalid strike price	26
Invalid expiry date	27
Invalid option type	28
Invalid trade quantity	29
Invalid trade price	30
Invalid order number	31
Invalid trade number	32
Invalid broker id	33
Already submitted	50
CP-CM association should be present	51
CP-CM association should not be changed	65

***** End of Document *****