腾讯云 IM 开放-WebSDK-关系链管理 REST 协议文档

文档编号		
文档名称	腾讯云 IM 开放-WebSdk-关系链管理 REST 协议文档	
作者	peakerdong	
首次完成时间	2015-08-13	
当前版本	V1.0	
修改记录		
时间	修改人	修改说明

目 录

1 综述

腾讯云 IM 开放-Web SDK-关系链管理 REST 包含以下功能：

- ✓ 多种模式的批量添加好友（包括一回合添加、两回合添加等等）；
- ✓ 多种模式的批量删除好友（包括单向删除、双向删除等等）；
- ✓ 多种模式的好友拉取（包括不带好友的增量模式、全量分页方式以及带好友方式等等）；
- ✓ 批量拉取好友申请；
- ✓ 批量删除好友申请；
- ✓ 批量添加黑名单；
- ✓ 批量删除黑名单；
- ✓ 拉取黑名单；

2 名词解释

- ❖ 昵称：用户为自己设置的名称，比如 QQ 用户可以在客户端设置自己的昵称。
- ❖ 加好友验证方式：用户可以选择自己以哪种方式被加好友，比如可以选择“别人添加我的时候需要我的验证”，或者“我允许任何人添加我不需要验证”。
- ❖ 资料：一组描述用户属性的数据，目前用户的资料包括昵称，加好友验证方式。
- ❖ 好友表：用户的好友列表。
- ❖ 黑名单：用户的黑名单列表。
- ❖ 关系链：一组描述用户和其他用户关系的数据，目前用户的关系链包括好友表、黑名单。
- ❖ 一回合加好友：如果帐号 A 设置的加好友验证方式是允许任何人，那么任何人添加 A 为好友都可以直接添加成功，这种一个请求就添加好友成功的场景称作一回合加好友。
- ❖ 两回合加好友：如果帐号 A 设置的加好友验证方式是需要验证，那么任何人添加 A，A 都会收到一个请求加好友验证消息，这是第一个回合，然后 A 对这个请求加好友验证消息进行同意操作时，这是第二个回合，所以此类场景成为两回合加好友。

3 REST 协议说明

目前 web sdk 提供以下几个跟关系链有关的 api 接口：

API 名字	协议功能简述
applyAddFriend	申请增加好友
responseFriend	响应加好友请求
deleteFriend	删除好友
getAllFriend	拉取好友表
getPendency	拉取好友申请
deletePendency	删除好友申请
addBlackList	增加黑名单
deleteBlackList	删除黑名单
getBlackList	拉取黑名单

3.1 申请增加好友

API 名称	applyAddFriend
请求参数示例	{ "From_Account":"id", "AddFriendItem": [{ "To_Account":"id1", "Remark":"remark1", "AddSource":"AddSource_Type_Unknow", "AddWording":"请加我为好友，我是你好友" }, { "To_Account":"id2",

<table>
<tr><td></td><td>"Remark":"remark2",
"AddSource":"AddSource_Type_Unknow",
"AddWording":"请加我为好友，我是你好友"
},
{
"To_Account":"id3",
"Remark":"remark3",
"AddSource":"AddSource_Type_Unknow",
"AddWording":"请加我为好友，我是你好友"
}
]
}</td></tr>
<tr><td>返回结果示例</td><td>{
"ResultItem":
[
{
"To_Account":"id1",
"ResultCode":0,
"ResultInfo":""
}
],
"Fail_Account":["id2"],
"Invalid_Account":["id3"],
"ActionStatus":"FAIL",
"ErrorCode":10001,
"ErrorInfo":"Err_SNS_FriendAdd_Msg",
"ErrorDisplay":""
}</td></tr>
<tr><td>接口说明</td><td>✓ 这个请求的语义是：以帐号 id 的身份来发起添加帐号名为 id1、id2、id3 的用户为好友。</td></tr>
</table>

	✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ Remark：添加对方时设置的备注名称，UTF8 格式，最长为 96 字节。 ✓ AddSource：添加好友时的来源，目前唯一的有效值是“AddSource_Type_Unknow”。 ✓ AddWording：添加好友时，对好友的附言信息，UTF8 格式，最长为 120 字节。 ✓ Fail_Account：如果某个帐号的 ResultCode 不为 0，那么就会在这个地方记录一次，Fail_Account 的信息和包体里面 ResultCode 字段是冗余的，这里只是为了提供调用的遍历性，调用方不需要逐个遍历 ResultCode，只要检查 Fail_Account 就可以知道哪些账户失败了。 ✓ Invalid_Account：如果调用的请求包里面所携带的 To_Account 中有非法帐号，比如注销过的账户等，就会把对应的 To_Account 放到 Invalid_Account 中。 ✓ ErrorCode 的意义请参考错误码描述章节部分。 ✓ 加好友提供批量添加的协议，如果对方设置的需要验证的方式，那么只会给对方添加一个未决请求，如果对方设置的是允许任何人的方式，那么会双向添加成好友。

3.2 响应加好友请求

API 名称	responseFriend
请求参数示例	{ "From_Account":"id", "ResponseFriendItem": [{ "To_Account":"id1", "Remark":"remark1", "ResponseAction":"Response_Action_Agree" }, { "To_Account":"id1", "Remark":"remark2",

<table>
<tr><td></td><td>"ResponseAction":"Response_Action_AgreeAndAdd"
},
{
"To_Account":"id3",
"Remark":"remark3",
"ResponseAction":"Response_Action_AgreeAndAdd"
}
]
}</td></tr>
<tr><td>返回结果示例</td><td>{
"ResultItem":
[
{
"To_Account":"id1",
"ResultCode":0,
"ResultInfo":""
}
],
"Fail_Account":["id2"],
"Invalid_Account":["id3"],
"ActionStatus":"FAIL",
"ErrorCode":10001,
"ErrorInfo":"Err_SNS_FriendResponse_Msg",
"ErrorDisplay":""
}</td></tr>
<tr><td>接口说明</td><td>✓ 这个请求的语义是：以帐号 id 的身份来响应 id1、id2、id3 发起的加好友请求。
✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。
✓ Remark：添加对方时设置的备注名称，UTF8 格式，最长为 96 字节。
✓ ResponseAction：响应方式，有且仅有以下三种方式：</td></tr>
</table>

	仅同意：Response_Action_Agree 同意并添加：Response_Action_AgreeAndAdd 拒绝：Response_Action_Reject。 ✓ Fail_Account：如果某个帐号的 ResultCode 不为 0，那么就会在这个地方记录一次，Fail_Account 的信息和包体里面 ResultCode 字段是冗余的，这里只是为了提供调用的遍历性，调用方不需要逐个遍历 ResultCode，只要检查 Fail_Account 就可以知道哪些账户失败了。 ✓ Invalid_Account：如果调用的请求包里面所携带的 To_Account 中有非法帐号，比如注销过的账户等，就会把对应的 To_Account 放到 Invalid_Account 中。 ✓ ErrorCode 的意义请参考错误码描述章节部分。

3.3 删除好友

API 名称	deleteFriend

请求参数示例	{
	"From_Account":"id",
	"To_Account":["id1","id2","id3"],
	"DeleteType":"Delete_Type_Both"
	}

返回结果示例	{
	"ResultItem":
	[
	{
	"To_Account":"id1",
	"ResultCode":0,
	"ResultInfo":""
	}
	],
	"Fail_Account":["id2"],
	"Invalid_Account":["id3"],
	"ActionStatus":"FAIL",

	"ErrorCode":10001, "ErrorInfo":"Err_SNS_deleteFriend_Msg", "ErrorDisplay":"" }
接口说明	✓ 这个请求的语义是：以帐号 id 的身份来发起删除 id 好友表中帐号名为 id1、id2、id3 的好友。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ DeleteType：删除方式，有且仅有以下两种取值： 单项删除：Delete_Type_Single（仅在 From_Account 里面删除 To_Account） 双向删除：Delete_Type_Both（在 From_Account 里面删除 To_Account，同时在 To_Account 中也删除 From_Account）。 ✓ Fail_Account：如果某个帐号的 ResultCode 不为 0，那么就会在这个地方记录一次，Fail_Account 的信息和包体里面 ResultCode 字段是冗余的，这里只是为了提供调用的遍历性，调用方不需要逐个遍历 ResultCode，只要检查 Fail_Account 就可以知道哪些账户失败了。 ✓ Invalid_Account：如果调用的请求包里面所携带的 To_Account 中有非法帐号，比如注销过的账户等，就会把对应的 To_Account 放到 Invalid_Account 中。 ✓ ErrorCode 的意义请参考错误码描述章节部分。

3.4 拉取好友列表

API 名称	getAllFriend
请求参数示例	{ "From_Account":"id", "TimeStamp":1429000001, "StartIndex":0, "TagList": ["Tag_Profile_IM_Nick", "Tag_SNS_IM_Remark"

	], "LastStandardSequence":1, "GetCount":100 //拉取的数目 }

返回结果示例	{
	"NeedUpdateAll":"GetAll_Type_NO",
	"TimeStampNow":1430000001,
	"StartIndex":0,
	"InfoItem":
	[
	{
	"Info_Account":"id1",
	"SnsProfileItem":
	[
	{
	"Tag":"Tag_SNS_IM_Remark",
	"Value":"RemarkNameTest1"
	}
	]
	},
	{
	"Info_Account":"id2",
	"SnsProfileItem":
	[
	{
	"Tag":"Tag_SNS_IM_Remark",
	"Value":"RemarkNameTest2"
	}
	]
	}

	], "CurrentStandardSequence":2, "FriendNum": 1000, "ActionStatus":"FAIL", "ErrorCode":10001, "ErrorInfo":"Err_SNS_getAllFriend_Msg", "ErrorDisplay":"" }
接口说明	✓ 这个请求的语义是：拉取帐号 id 的好友表，上次拉取好友表是发生 1429000001 秒（时间基准点是 1970-01-01 00:00:00 UTC）,客户端本次缓存的好友表数据版本号是 1，需要拉取好友的昵称和我对好友的备注，并且是从好友表的最开始分页拉取好友，每次返回 100 个好友给客户端。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ TimeStamp：上次拉取好友表的时间戳，如果是第一次拉取，那么 TimeStamp 填 0，客户端在拉取时带上此时间戳，告知服务端“我要拉取好友表，以及好友的某个字段”，服务端会将此时间戳和服务器端针对某个字段的更新时间戳进行比较，如果服务器端的时间戳不大于客户端的时间戳，那么就认为客户端拥有最新的数据，此时不会返回数据给客户端，否则返回数据给客户端。如果客户端每次都希望重新拉取，那么 TimeStamp 每次请求时都填为 0。 ✓ StartIndex（请求包中）：好友表的起始位置，如果填 0 表示从好友表的最前面开始拉取，如果填 100 则表示从好友表升序排序的第 100 个位置开始拉取。 ✓ TagList：需要拉取的字段，该拉取协议是一条整合数据的协议，可以指定拉取自己好友的昵称、加好友设置以及对用户的备注等字段，如果需要拉取昵称字段，则这里就需要在 Json 数组中填入 Tag_Profile_IM_Nick。 ✓ LastStandardSequence：用户好友表数据的版本号，如果用户修改了自己的好友表那么在服务器端就给该 Sequence 自增一次。因此当请求中有资料类的 Tag 时（比如 Tag_Profile_IM_Nick），LastStandardSequence 会自动失效，服务器端忽略请求中的 LastStandardSequence，并且认为客户端携带上来的 LastStandardSequence 为 0，也就是说 LastStandardSequence 仅在请求中只拉取好友表数据时有效。客户端第一次

	拉取时，LastStandardSequence 填写为 0，并且记录本次拉取好友表的回包里面的 CurrentStandardSequence，以便下次拉取作为 LastStandardSequence 来发起调用，如果客户端带上来的 LastStandardSequence 和服务器端保存的 Sequence 相同，那么就不返回数据给客户端，因为服务器认为客户端拥有上一次的历史数据，如果客户端带上来的 LastStandardSequence 和服务器端保存的 Sequence 不相同，那么就返回数据，因为服务器认为客户端的历史数据已经不是最新的了，有必要给客户端下发最新的好友表。 ✓ GetCount：分页拉取时每次拉取多少个好友。 ✓ NeedUpdateAll：是否需要进行全量拉取，有且仅有以下两种取值： 不需要全量拉取：GetAll_Type_NO 需要全量拉取：GetAll_Type_YES。 ✓ TimeStampNow：服务器端返回的时间戳，下次拉取好友表时将该时间戳作为请求包中的 TimeStamp 带上给服务器端。 ✓ StartIndex（回包中的）：下次分页拉取的起始位置，如果分页拉取完毕了，服务器端返回 0 给客户端，客户端需要判断返回的 StartIndex 为 0 就结束分页拉取。 ✓ CurrentStandardSequence：用户好友表数据的最新版本号，客户端需要将此值保存下来下次请求时带上给服务器端。 ✓ FriendNum：帐号 id 的好友表个数。 ✓ ErrorCode 的意义请参考错误码描述章节部分。 ✓ 拉取好友表协议提供了增量拉取模式和全量拉取模式，当 TimeStamp 为 0 时进入全量拉取模式，此时一般情况下全量拉取的第一个请求需要将 LastStandardSequence 和 StartIndex 均设置为 0，全量拉取的第二个分页请求则将第一个请求回包中的 StartIndex 带上继续向服务器端访问，当 TimeStamp 不为 0 时进入增量模式，此时如果请求包中仅拉取关系链数据，则根据 LastStandardSequence 是否与服务器端的 Sequence 相等来判断是否要返回数据，如果请求包中除了拉取关系链数据外，也拉取其他数据，那么判断 LastStandardSequence 的逻辑将会失效。

3.5 拉取好友申请

API 名称	getPendency
请求参数示例	{

<table>
<tr><td></td><td>"From_Account":"id",
"PendencyType":"Pendency_Type_ComeIn",
"StartTime":0,
"MaxLimited":10,
"LastSequence":1
}</td></tr>
<tr><td>返回结果示例</td><td>{
"PendencyItem":
[
{
"To_Account":"id1",
"AddTime":1430000008,
"AddSource":"Futu_Add_From_PC",
"AddWording":"你好，我是你的朋友",
"Nick":"id1_nickname"
},
{
"To_Account":"id2",
"AddTime":1430000009,
"AddSource":"Futu_Add_From_PC",
"AddWording":"你好，我是你的朋友",
"Nick":"id2_nickname"
}
],
"StartTime":1430000010,
"UnreadPendencyCount":0,
"CurrentSequence":2,
"ActionStatus":"FAIL",
"ErrorCode":10001,
"ErrorInfo":"Err_SNS_PendencyGet_Msg",</td></tr>
</table>

	"ErrorDisplay":"" }
接口说明	✓ 这个请求的语义是：拉取帐号 id 的未处理的好友请求。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ PendencyType：请求类型，有且仅有以下两种取值： 别人发给我的："Pendency_Type_ComeIn", 我发给别人的："Pendency_Type_SendOut" ✓ StartTime：好友申请的起始时间。 ✓ MaxLimited：分页大小，取值为 30 表示客户端要求服务器端每页最多返回 30 个好友申请。 ✓ LastSequence：好友申请数据的版本号，用户每收到删除一条好友申请，服务器端就自增一次好友申请数据版本号，客户端第一次拉取时，LastSequence 填写为 0，并且记录本次拉取好友申请的回包里面的 CurrentSequence，以便下次拉取作为 LastSequence 来发起调用，如果客户端带上来的 LastSequence 和服务器端保存的 Sequence 相同，那么就不返回数据给客户端，因为服务器认为客户端拥有上一次的历史数据，如果客户端带上来的 LastSequence 和服务器端保存的 Sequence 不相同，那么就返回数据，因为服务器认为客户端的历史数据已经不是最新的了，有必要给客户端下发最新的好友申请。 ✓ CurrentSequence：好友申请数据的最新版本号，客户端需要将此值保存下来下次请求时带上给服务器端。 ✓ ErrorCode 的意义请参考错误码描述章节部分。

3.6 删除好友申请

API 名称	deletePendency
请求参数示例	{ "From_Account":"id", "To_Account":["id1","id2"]

	"PendencyType":"Pendency_Type_ComeIn" }

返回结果示例	{
	"ResultItem":
	[
	{
	"To_Account":"id1",
	"ResultCode":0,
	"ResultInfo":""
	}
	],
	"Invalid_Account":["id2"],
	"ActionStatus":"FAIL",
	"ErrorCode":10001,
	"ErrorInfo":"Err_SNS_PendencyDelete_Msg",
	"ErrorDisplay":""
	}

接口说明	✓ 这个请求的语义是：以帐号 id 的身份删除来自账号名为 id1、id2、id3 的用户的好友请求。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ PendencyType：请求类型，有且仅有以下两种取值： 别人发给我的："Pendency_Type_ComeIn", 我发给别人的："Pendency_Type_SendOut" ✓ Fail_Account：如果某个帐号的 ResultCode 不为 0，那么就会在这个地方记录一次，Fail_Account 的信息和包体里面 ResultCode 字段是冗余的，这里只是为了提供调用的遍历性，调用方不需要逐个遍历 ResultCode，只要检查 Fail_Account 就可以知道哪些账户失败了。 ✓ Invalid_Account：如果调用的请求包里面所携带的 To_Account 中有非法帐号，比如注销过的账户等，就会把对应的 To_Account 放到 Invalid_Account 中。

	✓ ErrorCode 的意义请参考错误码描述章节部分。
	✓

3.7 增加黑名单

API 名称	addBlackList

请求参数示例	{
	"From_Account":"id",
	"To_Account":["id1","id2","id3"]
	}

返回结果示例	{
	"ResultItem":
	[
	{
	"To_Account":"id1",
	"ResultCode":0,
	"ResultInfo":""
	}
	],
	"Fail_Account":["id2"],
	"Invalid_Account":["id3"],
	"ActionStatus":"FAIL",
	"ErrorCode":10001,
	"ErrorInfo":"Err_SNS_addBlackList_Msg",
	"ErrorDisplay":""
	}

接口说明	✓ 这个请求的语义是：以帐号 id 的身份来发起添加帐号名为 id1、id2、id3 的用户为黑名单。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ Fail_Account：如果某个帐号的 ResultCode 不为 0，那么就会在这个地方记录一次，

	Fail_Account 的信息和包体里面 ResultCode 字段是冗余的，这里只是为了提供调用的遍历性，调用方不需要逐个遍历 ResultCode，只要检查 Fail_Account 就可以知道哪些账户失败了。 ✓ Invalid_Account：如果调用的请求包里面所携带的 To_Account 中有非法帐号，比如注销过的账户等，就会把对应的 To_Account 放到 Invalid_Account 中。 ✓ ErrorCode 的意义请参考错误码描述章节部分。

3.8 删除黑名单

API 名称	deleteBlackList

请求参数示例	{
	"From_Account":"id",
	"To_Account":["id1","id2","id3"]
	}

返回结果示例	{
	"ResultItem":
	[
	{
	"To_Account":"id1",
	"ResultCode":0,
	"ResultInfo":""
	}
	],
	"Fail_Account":["id2"],
	"Invalid_Account":["id3"],
	"ActionStatus":"FAIL",
	"ErrorCode":10001,
	"ErrorInfo":"Err_SNS_deleteBlackList_Msg",
	"ErrorDisplay":""
	}

接口说明	✓ 这个请求的语义是：以帐号 id 的身份将帐号名为 id1、id2、id3 的用户从黑名单中

	移除。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ Fail_Account：如果某个帐号的 ResultCode 不为 0，那么就会在这个地方记录一次，Fail_Account 的信息和包体里面 ResultCode 字段是冗余的，这里只是为了提供调用的遍历性，调用方不需要逐个遍历 ResultCode，只要检查 Fail_Account 就可以知道哪些账户失败了。 ✓ Invalid_Account：如果调用的请求包里面所携带的 To_Account 中有非法帐号，比如注销过的账户等，就会把对应的 To_Account 放到 Invalid_Account 中。 ✓ ErrorCode 的意义请参考错误码描述章节部分。

3.9 拉取黑名单

API 名称	getBlackList

请求参数示例	{
	"From_Account":"id",
	"StartIndex":0,
	"MaxLimited":30,
	"LastSequence":1
	}

返回结果示例	{
	"BlackListItem":
	[
	{
	"Black_Account":"id1",
	"BlackTimeStamp":1430000001
	},
	{
	"Black_Account":"id2",
	"BlackTimeStamp":1430000002
	}

	], "StartIndex":2, "CurrentSequence":2, "ActionStatus":"OK", "ErrorCode":0, "ErrorInfo":"", "ErrorDisplay":"" }
接口说明	✓ 这个请求的语义是：拉取帐号为 id 的用户的黑名单。 ✓ From_Account：请求发起方的帐号，一般情况下为用户本人帐号，在管理员代替用户发起请求的情况下，这里应该填写被代替的用户的帐号。 ✓ StartIndex（请求包中的）：黑名单的起始位置，如果填 0 表示从黑名单的最前面开始拉取，如果填 100 则表示从黑名单升序排序的第 100 个位置开始拉取。 ✓ MaxLimited：分页大小，取值为 30 表示客户端要求服务器端每页最多返回 30 个黑名单。 ✓ LastSequence：黑名单数据的版本号，用户每增加/删除一次黑名单，服务器端就自增一次黑名单数据版本号，客户端第一次拉取时，LastSequence 填写为 0，并且记录本次拉取黑名单的回包里面的 CurrentSequence，以便下次拉取作为 LastSequence 来发起调用，如果客户端带上来的 LastSequence 和服务器端保存的 Sequence 相同，那么就不返回数据给客户端，因为服务器认为客户端拥有上一次的历史数据，如果客户端带上来的 LastSequence 和服务器端保存的 Sequence 不相同，那么就返回数据，因为服务器认为客户端的历史数据已经不是最新的了，有必要给客户端下发最新的黑名单。 ✓ StartIndex（回包中的）：下次分页拉取的起始位置，如果分页拉取完毕了，服务器端返回 0 给客户端，客户端需要判断返回的 StartIndex 为 0 就结束分页拉取。 ✓ CurrentSequence：黑名单数据的最新版本号，客户端需要将此值保存下来下次请求时带上给服务器端。 ✓ BlackTimeStamp：加黑名单的时间戳。 ✓ ErrorCode 的意义请参考错误码描述章节部分。

4 错误码说明

普通错误码说明中的错误码需要您进行相应地处理，而异常错误码说明中的错误码则不需要您关注，只要您注意到这是一个错误就可以了。

4.1 普通错误码说明

命令名	错误码	错误描述
Common(通用)	30001	关系链解包失败
	30002	SDKAppId 非法
applyAddFriend	30501	加好友来源参数错误
	30502	加好友参数长度错误
	30503	加好友个数错误
	30515	已设置被加好友为黑名单
	30516	被加好友设置禁止被添加
	30517	未决一次被加满
	30518	加好友组满
	30519	加好友好友满
	30520	加好友好友已经存在
	30521	获取好友资料信息不存在
	30525	已被被加好友设置为黑名单
	30535	添加对方好友信息好友满
	30539	加未决成功(用特殊错误码特殊标示和好友加成功区分)
responseFriend	30601	加好友回应动作参数错误
	30602	加好友回应参数长度错误
	30603	加好友回应好友个数错误
	30614	加好友回应未决不存在
	30615	加好友回应己方好友满
	30616	加好友回应组满
	30617	加好友回应好友已经存在
	30630	加好友回应对方好友满
getAllFriend	31101	拉取所有好友参数错误
	31102	拉取所有好友时字段数目异常
	31103	拉取所有好友主键不存在

addBlackList	31301 31302 31306 31307	请求添加的黑名单个数非法 校验黑名单的 SDKAppId 失败 添加黑名单失败 待添加的 TinyId 在黑名单里
deleteBlackList	31501 31503 31504	请求删除的黑名单个数非法 请求删除不存在的黑名单 删除黑名单失败
getBlackList	31601 31602	请求拉取的黑名单个数非法 拉取黑名单失败
deleteFriend	31701 31702 31703 31704	删除好友的请求个数非法 删除好友的请求类型非法 校验正向的标配数据失败 请求删除的号码不是好友

4.2 异常错误码说明

命令名	错误码	错误描述
applyAddFriend	30504	加好友获取 SDKAppId 失败
	30505	加好友执行任务失败
	30506	获取好友列表错误
	30507	获取好友列表元数据错误
	30508	拉取好友列表行数据错误
	30509	添加己方好友信息行数据错误
	30510	添加己方好友信息元数据错误
	30511	添加己方好友信息错误
	30512	删除己方未决错误
	30513	拉取黑名单列表错误
	30514	添加好友新建任务失败
	30522	获取好友资料信息错误
	30523	获取好友资料信息行数据错误
	30524	获取被加好友的的黑名单错误
	30526	获取对方和己方的好友关系错误

	30527	获取对方和己方的好友关系行数据错误
	30528	添加对方好友信息时获取序列号错误
	30529	添加对方好友信息时获取序列号元数据错误
	30530	添加对方好友信息时获取未决错误
	30531	添加对方好友信息时获取未决行数据错误
	30532	添加对方好友信息时删除未决失败
	30533	添加对方好友信息时获取好友数目数目
	30534	添加对方好友信息时获取好友数目序列号不一致
	30536	添加对方好友信息元数据错误
	30537	添加对方好友信息行数据错误
	30538	添加对方好友信息错误
responseFriend	30604	加好友回应获取未决错误
	30605	加好友回应获取未决元数据错误
	30606	加好友回应获取未决行数据错误
	30607	加好友回应获取好友数错误
	30608	加好友回应获取好友数序列号变化
	30609	加好友回应设置己方好友信息行数据错误
	30610	加好友回应设置己方好友信息元数据错
	30611	加好友回应设置己方好友信息错误
	30612	加好友回应设置己方删除未决错误
	30613	加好友回应执行任务失败
	30618	加好友回应设置对方好友信息时检查好友存在失败
	30619	加好友回应设置对方好友信息时检查好友存在元数据错误
	30620	加好友回应设置对方好友信息时检查好友存在行数据错误
	30621	加好友回应设置对方好友信息时获取好友数失败
	30622	加好友回应设置对方好友信息时获取好友数序列号变化
	30623	加好友回应设置对方好友信息时添加好友元数据错误
	30624	加好友回应设置对方好友信息行添加好友行数据错误
	30625	加好友回应设置对方好友信息时添加好友错误
	30626	加好友回应设置对方好友信息时删除未决错误
	30627	加好友回应设置对方好友信息时设置序列号元数据错误
	30628	加好友回应设置对方好友信息时设置序列号错误
	30629	加好友回应新建任务失败
getAllFriend	31104	拉取所有好友获取序列号错误
	31105	拉取所有好友获取序列号元数据错误
	31106	拉取所有好友获取字段对应的字符串错误

	31107	拉取所有好友关系链标配信息错误
	31108	拉取所有好友关系链标配信息组数据错误
	31109	拉取所有好友关系链标配信息行数据错误
	31110	拉取所有好友关系链自定义信息错误
	31111	拉取所有好友关系链自定义信息行数据错误
	31112	拉取所有好友执行任务失败
addBlackList	31303	校验正向标配好友关系失败
	31304	删除正向标配好友关系失败
	31305	校验黑名单失败
deleteBlackList	31502	校验黑名单失败
deleteFriend	31705	删除正向的标配数据失败
	31706	删除反向数据失败
getPendency	31001	获取未决参数错误
	31002	获取的好友来源错误
	31003	获取未决时主键不存在
	31004	获取未决时序列号错误
	31005	获取未决时序列号元数据错误
	31006	获取未决错误
	31007	获取未决元数据错误
	31008	获取未决昵称执行任务失败
deletePendency	31801	删除未决的请求个数非法
	31802	删除未决的请求类型非法
	31803	校验正向的未决关系失败
	31804	请求删除的号码不是未决
	31805	删除正向的标配数据失败