Open Communications Platform (OCP) PHP API Guide and Reference

Version 1.1

950 Tower Lane
Foster City, CA 94404
T: 650-525-9200
F:650.287.2628
www.intelepeer.com

Copyright

Table of Contents

Using the OCP Platform

The IntelePeer Open Communications Platform (OCP) is an application programming interface (API) that allows programmers to embed voice telephony in Web and business applications. This document introduces you to the programming environment, its concepts, and terminology. The documentation also provides a comprehensive reference for each public class and function available in the API.

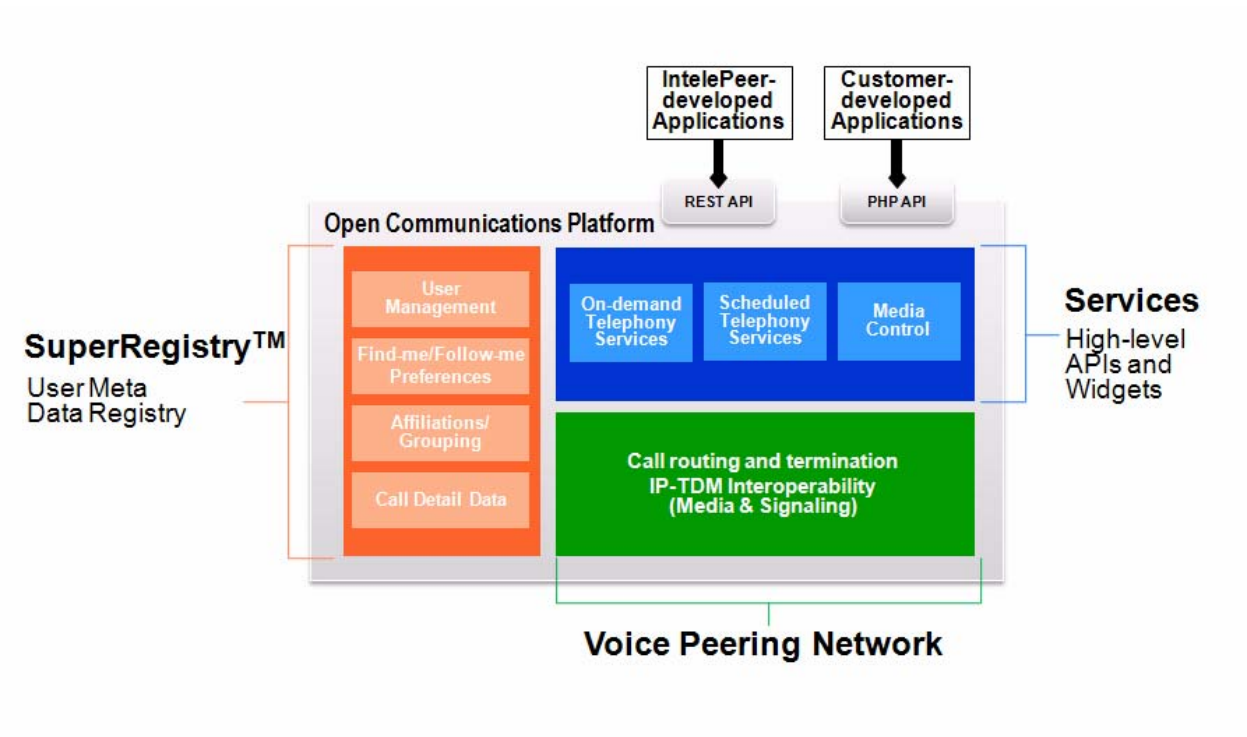
About this Documentation

You should read this document if you are interested in programming with PHP. If you would like to use the REST interface, contact your IntelePeer representative for a copy of that documentation.

Throughout this document, the names of application interfaces, their attributes, and functions appear in `courier` font.

What is the Open Communications Platform (OCP)?

The IntelePeer OCP API is a suite of telephony services encompassing auto conferencing, click-to-call, blast (group voice reminders), and conference-on-demand. These services reside on top of and make use of IntelePeer's carrier-grade infrastructure. The following diagram depicts the platform:



The API also includes user management and accounting functions. With these functions your application can easily do any of the following:

- create, edit, and delete users

- set and adjust user find-me /follow-me preferences
- ensure private/anonymous calling
- manage phone numbers
- set and edit minutes-of-use (MOU) data
- set and edit incurred charges on a user

Most OCP services require that a user exist in the IntelePeer database, but not all of them. However, use of the OCP itself requires your organization to have a valid Enterprise Identifier (EID). IntelePeer assigns this EID after a partnership agreement has been reached.

How you secure your application depends on your environment. If you are working on a private network, you can use private network connectivity to secure your application. If your application resides on a public network you can use https.

Managing Users

This section is an overview of the classes and functions you will use to work with users. The OCP user management features allow you to create a user, manage the user's find-me/follow-me preferences, and set up accounting for minutes-of-use and currency.

Creating a User

The **`User`** class represents all the non-phone attributes associated with a human being in your application. Creating a user requires you to specify the following information:

- first name
- last name
- email address
- default phone number

The email address and the phone number you specify must be unique within the IntelePeer system. You pass in a **`PhoneNumber`** object for the default number.

When you call **`userCreate()`** and pass in an instance of this class, the system creates a user, gives it an ID, and adds it to the IntelePeer database. Your **`User`** object now corresponds to an IntelePeer system user. Most functions in the API require you to supply a user's **`UID`**.

A **`PhoneNumber`** can represent any number belonging to any carrier any where in the world. Currently, however, the OCP telephony services can only make calls to the United States and Canada. A user can have multiple phone numbers where they might wish to be contacted such as home, office, cell phone, and so forth.

Calling Preferences

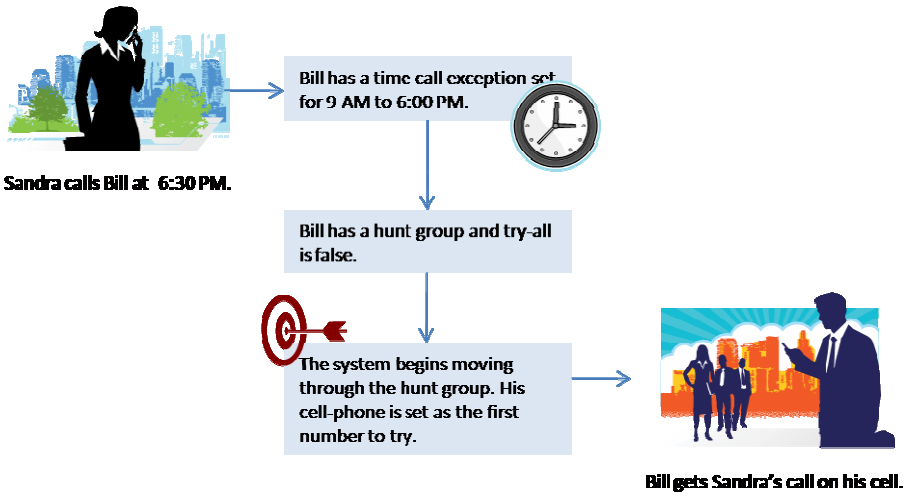
Users will have preferences regarding when they will accept calls on a particular phone. You use **`CallTimeException`** objects to describe specific time periods. For example, your application can allow

users to specify a block of time such as Monday through Friday 9 AM to 5 PM when they prefer to receive calls on their office phone. You use **`setCallTimeException()`** to associate these time periods with a particular number belonging to a user.

You can also define a hunt group for a user. A hunt group orders the user's phone numbers from first to last. When a call is made to the user, the system begins at the first number and calls each number in turn until either a human being answers or it reaches the last number.

Hunt groups also include a try-all attribute that instructs the system to try-all the user's numbers in the group when connecting a call. If this option is set, the system rings all the user's phones ***simultaneously***. The first number to be answered by a human voice, receives the call.

A system considers a user's calling preferences when attempting to reach a user with a call. The following diagram depicts a typical scenario and how the system progresses as it interprets user preferences:



When calling a user, the system always first uses a user's call-time exception preferences. Then, the system attempts to contact a user through a hunt group. If the user has no hunt group set, the system uses the default number on the user. If a hunt group exists, the system considers the try-all attribute, calling all numbers simultaneously if this option is true. If try-all is false, the system attempts to contact the user by sequentially trying each number in the hunt group.

A user can also set a DND preference. This allows a user to set blocks of time in which all calls are blocked.

Contact Groups and Black Lists

A particular user may have a set of numbers that he or she calls frequently. For example, Bill calls Sandra his coworker, Alice his wife, his son Todd, and his running buddy Zhao. Your application can allow Bill to create one or more set of contact groups. For example, Bill might set up a group for his family members. Then, when Bill is going to be late, he can send a blast call to his family group informing them.

Occasionally, a user might want to block one or more numbers. This is special type of group associated with a user called a black list. Numbers on the black list cannot use the IntelePeer system to dial the user. Your application can allow a user to add and remove numbers from a black list.

User Accounting: Minutes and Charges

Your application establishes each user with a minutes-of-use (MOU) value using the **`setMOUAllowed()`** function. The OCP keeps track of user calls and increments the user's MOU value when a call is made. You can use the **`getMOUused()`** function to get a user's current MOU. If you wish to set MOU limits, your application is responsible for ensuring a user does not exceed their MOU limit. For example, before placing a call your application can use **`getMOUused()`** and compare it to **`setMOUAllowed()`** to check if a user has the MOU available to make a call.

Similar to MOU, you can attach a currency value (charge) to a user using the **`setUserCurrency()`** function. The OCP has the **`getUserCurrency()`** function, **`incrementUserCurrency()`** function, and the **`decrementUserCurrency()`** function. Your application is responsible for using these functions to edit a user's currency in accordance with the schema your company defines. The denomination of currency is up to your application.

Making Calls with the OCP

The OCP allows you to write applications that make several types of phone calls. Before your application can make use of the OCP, it must use your company's enterprise ID and key to start a session with the **`requestSession()`** function. Once a session is established the system can manage users and make calls. The system supports the following call types:

`clickToCall()`	The system connects one point to another immediately.
`blastVoice()`	The system sends an announcement from one point to one or more recipients.
`scheduleAutoConference()`	A user elects to participate in a conference and the system joins that user to the conference when it occurs.
`conferenceOnDemand()`	A user, acting as a moderator, schedules a conference and invites one or more participants. The system joins the moderator and participants into a "conference room" on the IntelePeer system.

When your application creates a call, it supplies two **`EndpointIdentifier`** instances for both "legs" of the call — the origination point and the end point. When connecting two calls, the IntelePeer software always connects the end point (or points) first and then connects back to the call originator.

In all cases except for the **`blastVoice()`** call, the system considers it a success when a call reaches a human voice. If the system reaches a human voice, the system requests that the recipient press 1 to accept the incoming call. When it receives a positive response, the system acts appropriately according to the type of call:

`clickToCall()`	Connects back to the originator..

`scheduleAutoConference()`	Connects the recipient to the conference. The recipient need not know the conference connection information.
`conferenceOnDemand()`	Connects the recipient to the conference. The recipient need not know the conference connection information.

A **`blastVoice()`** call considers an answering machine response a success as well as a human voice. When a blast reaches a successful answer, it plays the blast message.

The system also has functions for examining the status of a scheduled call and the history of a completed call. Your application can, for example, get a list of the call identifiers that participated in a blast or a conference.

OCP Supporting Classes

You will use the classes in this section with the various Open Communications Platform (OCP) function calls. The following classes appear in this section:

`EndpointIdentifier`	Supplies origination or termination points for a telephone call. This class can represent a system user, an email address, a simple phone number or a contact group.
`User`	Defines non-phone attributes associated with a user. For example, this class contains the addresses, both physical and email, for a user.
`CallTimeException`	Represents a block of time for which a user would like to define a contact function. For example, a user may have different contact preferences for days than for evening hours.
`PhoneNumber`	Specifies the phone number for contacting a user. A system user may have one or more phone numbers.
`File`	Describes a file for use with a blast audio call.

EndpointIdentifier

Supplies origination or termination information for telephone dialing.

Syntax

```
class EndpointIdentifier
{
  // Type Definitions
  const ID_TYPE_UNKNOWN  = 0;
  const ID_TYPE_UID   = 1;
  const ID_TYPE_EMAIL   = 2;
  const ID_TYPE_PHONE   = 3;
  const ID_TYPE_GROUP   = 4;

  // Class Variables
  var $mId;
  var $mIdType;

  // Function Syntaxs
  function __construct($mId=0,$mIdType=0,&$errorText);
  function getType();
  function getId();
  function setType($type,&$errorText);
  function setId($id,&$errorText);
}
```

Description

The **EndpointIdentifier** class represents an endpoint in a call — either the point of origination or termination. The constructor takes a **mId** representing a value of type **mIdType**. An **EndpointIdentifier** object can be any of the following types:

ID_TYPE_UID — A user identifier in the IntelePeer database. When specifying a termination **EndpointIdentifier**, the system will resolve the user ID to **userInfo** object and use the user's call preferences.

ID_TYPE_EMAIL — An email address. For valid email addresses, the system attempts to resolve the email to a user ID in the IntelePeer database. If a user exists with the corresponding email, the system uses the user's call preferences.

ID_TYPE_PHONE — A phone number. If the **EndpointIdentifier** is a termination point and belongs to a user ID in the IntelePeer database, the system ignores the user preferences and rings the specified phone number directly

Currently, the system supports phone numbers for the United States and Canada only.

ID_TYPE_GROUP — A contact group associated with a specific user.

You can use the getters and setters on this class to manipulate object variables. Each function returns a 0 on success and a 1 on failure.

See Also

The functions **`clickToCall()`** on page 81, **`conferenceOnDemand()`** on page 93, **`blastVoice()`** on page 101.

User

Defines non-phone attributes associated with a system user.

Syntax

```
class User
{
  // Class Variables
  var $firstName;
  var $lastName;
  var $address1;
  var $address2;
  var $city;
  var $state;
  var $country;
  var $zip;
  var $gender;

  // Function Syntaxs
  function __construct();
  function setFirstName($value,&$errorText)
  function setLastName($value,&$errorText);
  function setAddress1($value,&$errorText);
  function setAddress2($value,&$errorText);
  function setCity($value,&$errorText);
  function setState($value,&$errorText);
  function setCountry($value,&$errorText);
  function setZip($value,&$errorText);
  function setGender($value,&$errorText);
  function getFirstName()
  function getLastName();
  function getAddress1();
  function getAddress2();
  function getCity();
  function getState();
  function getCountry();
  function getZip();
  function getGender();
}
```

Description

The **User** class contains the non-phone attributes associated with an IntelePeer system user. When creating a system user with **createUser()**, you pass a **User** object into the call. The object must contain at least a **firstName** and **lastName,** otherwise the call fails.

You can use the getters and setters on this class to manipulate its object variables. Each function returns a 0 on success and a 1 on failure.

CallTimeException

Defines a range of times.

Syntax

```
class CallTimeException
{
  // Type Definitions
  const SUN    = 1;
  const MON    = 2;
  const TUES   = 4;
  const WED    = 8;
  const THUR   = 16;
  const FRI    = 32;
  const SAT    = 64;

  // Class Variables
  var $mDaysInEffect;
  var $mExpiration;
  var $mStartHour;
  var $mEndHour;

  // Function Syntaxs
  function __construct();
  function setDaysInEffect($daysBitMask);
  function setStartHour ($startHour);
  function setStopHour($stopHour);
  function setExpiration($date);
  function getDaysInEffect();
  function getStartHour();
  function getStopHour();
  function getExpiration();
}
```

Description

The **CallTimeException** class describes a time period for use in user call preferences. For example, a user only wants to be reached at his work number from 9 to 5 every weekday. You can specify exceptions for specific hours or specific days-of-the week or a combination of both.

Specify hours in Greenwich Mean Time (GMT). You must specify contiguous clock times, for example, 8 AM to Noon. The expiration date is specified in Unix GMT time. Days are identified as a bit mask. You can OR together the values SUN, MON, TUES, and so forth to define multiple days. Days need not be contiguous, for example, THUR and SAT.

The **CallTimeException** class supports only a single range of contiguous time values. If a user has multiple non-contiguous time exceptions on a single day or across multiple days, the user's preferences must contain a **CallTimeException** for each time range.

When specifying call time exceptions, you are responsible for ensuring that incoming call time exceptions do not unintentionally overwrite any existing exceptions for the same time period. For example, if your

caller has preference already set for MON 8 AM to Noon and then sets a new preference for MON 7:30 AM to 1 PM, the new preference overwrites the old. If the new preference were for MON 10 to Noon, the user would then have two preferences for Monday, one from 8 AM to 10 and a new one from 10 to Noon.

See Also

The functions **`removeCallTimeException`** on page 47, **`getCallTimeExceptions`** on page 48.

PhoneNumber

Specifies a phone number to associate with a system user.

Syntax

```
class PhoneNumber
{
  // Type Definitions
  const OFFICE = 1;
  const MOBILE = 2;
  const HOME   = 3;
  const OTHER  = 4;

  // Class Variables
  var $mCountryCode=1;
  var $mPhoneId;
  var $mCityCode;
  var $mDigits;
  var $mPhoneType = OFFICE;
  var $isDefault=false;
  var $isSms=false;
  var $mCarrierId=0;
  var $validationStatus=2;

  // Function Syntaxs
  function __construct();
  function getId();
  function setDigits($digits,&$errorText);
  function setCountryCode($cc,&$errorText);
  function setCityCode($cc,&$errorText);
  function setPhoneType($value,&$errorText);
  function setWireless();
  function setAsDefault();
  function setAsSms();
  function setCarrierId($id,&$errorText);
  function getDigits();
  function setId($id,&$errorText);
  function setValidationStatus($status,&$errorText)
  function getDigits()
  function getCountryCode();
  function getCityCode();
  function getPhoneType();
  function isDefault();
  function isSms();
  function getCarrierId();
}
```

Description

The **PhoneNumber** class describes a phone number for a system user. When specifying a phone number, you must provide the proper format for domestic and international numbers in the **mDigits** field. If you are specifying a domestic number, simply enter the digits. For international calls, you must provide a

deliminating space between the phone number and the country and city code. The format for international numbers is as follows:

countrycode citycode phonenumber

A user can have multiple phone numbers. You can group phone numbers into a hunt group and associate it with a system user. You do not use **PhoneNumber** objects directly in functions that make calls. Instead, the **PhoneNumber** is placed in an **EndpointIdentifier** and the system uses the identifier to make the call.

You can specify that a number is an SMS number by setting the **isSms** value to true. By default, it is false. Use the **setAsSms()** function to set this value.

The **setAsDefault()** sets the default flag on the **PhoneNumber** object. To set the number as a user's default, use the **setDefaultNumber()** function.

Examples

The following is recognized by the API as a domestic number as it has 10 digits without spaces:

```
3035551212
```

The following number with a country code of 1 the API also recognizes as a domestic number:

```
1 3035551212
```

The following international number is for London. The country code is 44 = UK , city code 20 = London, the remainder the system recognizes as the phone's digits:

```
44 20 77994044
```

The following is a number in Paris (country code 33 = France, city code 1 = Paris):

```
33 1 49 54 46 46
```

The following are invalid international numbers. In the first, 4420 is an invalid country code and in the next it is more than 10 digits and there is no country code specified:

```
4420 77994044
33149544646
```

See Also

The **addUserPhones()** function on page 33.

File

Describes a file for use with a blast audio call.

Syntax

```
class File
{
  var $name;
  var $path;
  var $url;
  var $size;

  // Function Syntaxs
  function __construct($resource , $name='');
  function setName($name);
  function setPath($path);
  function setSize();
  function validFile();
  function cIVal($v);
  }
```

Description

The **File** class represents a file to upload or existing on the Intelepeer network. The system uses these files for blast audio calls. You create a file by specifying a file on your local system or an audio file on the Internet. The following are valid ways to create a new **File** object:

```
$myFile = new File('http://www.mysite.com/myfile.mp3');
$myFile = new File('http://www.mysite.com/myfile.mp3', 'myfile.mp3');
$myFile = new File('http://www.mysite.com/myfile.mp3',
'anEasierFilenameToRemember.mp3');
```

The file or the URL must be a MPEQ Audio Stream (**.mp3**) or Windows Media Audio (**.wma**) format. When a user uploads a file, the user provides a file name and the system saves this as the name but creates a temporary system-unique filename internally. The audio you want to blast cannot exceed 5 MB in size.

Variables

name	A string specifying the file's name as known to the user. Optional if you are specifying a URL with the constructor.; the system will take the name from the URL.
path	The path to the file on the local system.
url	The URL for a sound on the Internet.
size	The file size.

OCP Functions

This section contains the functions for managing users and making use of IntelePeer's services. Before you make these calls, you must use your enterprise ID and key to start a session. Each function has a set of variables. The system requires you to supply all the variables specified in the function's syntax.

All of the OCP functions return a value of 0 (zero) on success in addition to any other return values indicated. When a function fails, the system returns a non-zero error value and an **`errorText`**. The **`errorText`** contains the reason for the failure. This **`errorText`** may or may not be suitable for presentation to your end users.

requestSession()

Starts a session and returns its ID to the caller.

Syntax

```
function requestSession($enterpriseId,
                        $secret,
                        $gmtime
                        &$sessionId,
                        &$errorText)
```

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`Secret`	The key provided by IntelePeer.
`gmtime`	A Unix time stamp specifying when the system should place the call. If you do not specify a time, the system uses the current time and places the call immediately.

Description

`requestSession()` starts a new session for an application object and returns a session ID. Sessions expire after 24 hours. If your call returns a `SESSION_EXPIRED` error from an API call, it must request a new session before proceeding.

Returns

A `sessionId` and a 0 (zero) value. On failure, this function returns an `errorText` containing the reason for the failures and a non-zero value.

Errors

100	Incorrect timestamp in request
101	Unknown enterprise id
102	Invalid credentials
114	Internal error: Authentication system failed

generateSecret()

Generates a new random 25 character secret key.

Syntax

```
function generateSecret(&$secret,
                        &$errorText)
```

Description

generateSecret **()** generates a new random 25 character secret key..

Returns

A **secret key.** On failure, this function returns an **errorText** containing the reason for the failures and a non-zero value.

userCreate()

Creates a new system user.

Syntax

```
function userCreate($enterpriseId,
                    $sessionId,
                    $primaryEmail,
                    $defaultPhoneNumber,
                    $userInfo,
                    $exclusive=false,
                    &$UID,
                    &$errorText)
```

Description

userCreate() creates a system user and, upon success, returns a **UID** for the user. You use this UID in subsequent call management functions. The **userInfo** object you pass must include at least a **firstName** and **lastName** value. Finally, you must specify a unique the **primaryEmail** value and a **defaultPhoneNumber**.

By default, the system does not associate the user you create with your enterprise. This means that the user is accessible to all the enterprises on the IntelePeer OCP platform. To associate the user exclusively with your enterprise, set exclusive to **true**. This ensures that only applications with your **enterpriseId** can manipulate the user's call preferences or attributes.

☞ The system always associates a user's MOU and accounting values with a **enterpriseId** and **UID** combination regardless of how you set the **exclusive** value.

If the user already exists, the system returns the user's current information **UID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session ID as returned by **requestSession()**.
primaryEmail	A unique email address. Addresses that do not resolve return an error.
defaultPhoneNumber	A **PhoneNumber** object. The system uses this number as the user's default phone number. The default number can be bypassed by call exceptions or hunt sequences defined in the user preferences.
userInfo	A **User** object that must contain at least a **firstName** and a **lastName** value.
exclusive	A Boolean value specifying that the system should associate the user with the

enteripriseId.

Returns

Upon success, returns the **UID** assigned to this user and a 0 (zero). Upon failure, an **errorText** specifying the reason for the failure and a non-zero value.

Errors

103	User account already exists
105	Internal error: SQL execution failed

getPhoneId()

Gets the ID of a phone number.

Syntax

```
function getPhoneId($enterpriseId,
           $sessionId,
           $UID,
           $phoneNumber,
           &$phoneId,
           &$errorText,
           $addNumber = true)
```

Description

getPhoneId() search returns the **phoneId** of the specified **phoneNumber**. By default, if the phone number does not exist on the user, the system adds it to the user. Specify **addNumber = false** if you do not wish to add the number.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session ID as returned by **requestSession()**.
phoneNumber	A **PhoneNumber** object. This object must be unique.
addNumber	A Boolean value indicating whether or not to add the phone number. By default, this value is true.

Returns

Upon success, the **phoneId** value for the number and a 0 (zero). On failure, an **errorText** containing the reason for the failure and a non-zero value.

Errors

105	Internal error: SQL execution failed
110	Phone number not found

See Also

The **PhoneNumber** class on page 19.

updateUserPhone()

Gets the ID of a phone number.

Syntax

```
function updateUserPhone($enterpriseId,
           $sessionId,
           $UID,
           $oldPhone,
           $newPhone=null,
           $newPhoneType=null,
           &$errorText)
```

Description

updateUserPhone() changes the digits or phone type for existing numbers. This method does not change any of the calltime exceptions or rules associated with the phone.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session ID as returned by **requestSession()**.
UID	The user's ID. The system assigns this value when you call **createUser()**.
oldPhone	A **PhoneNumber** object.
newPhone	A **PhoneNumber** object. This object must be unique. Leave this blank if you wish to change only the type.
newPhoneType	You can set this value to **OFFICE**, **CELL**, **HOME**, or **OTHER**. By default, this value is null. Leave this blank if you only want to change the phone number.

Returns

On failure, an **errorText** containing the reason for the failure and a non-zero value.

Errors

105	Internal error: SQL execution failed
110	Phone number not found

See Also

The **`PhoneNumber`** class on page 19.

userDelete()

Deletes a user from the IntelePeer system.

Syntax

```
function userDelete($enterpriseId,
          $sessionId,
          $UID,
          &$errorText)
```

Description

userDelete() removes a user and its associated data from the IntelePeer system. Before deleting a user, your application is responsible for checking if the user (**UID**) is participating in any scheduled calls. Pending calls against the user's **UID** do not complete if you remove the user before the call. If a pending call is against the user's phone number, that call will complete.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session ID as returned by **requestSession()**.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, a 0 (zero). On failure, this function returns an **errorText** identifying the reason for the failure and a non-zero value.

Errors

105	Internal error: SQL execution failed
107	Incorrect email address for user
108	Cannot delete primary email address

getUserInfo()

Retrieves a **User** object.

Syntax

```
function getUserInfo($enterpriseId,
                     $sessionId,
                     $UID
                     &$userInfo,
                     &$errorText)
```

Description

getUserInfo() given a valid **UID** retrieves the **User** object associated with the ID. The **User** object contains only the non-phone attributes of the user. To get information regarding specific phone attributes such as a user's default phone number or calling preferences, you will need to use the specific getter functions for those values.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session ID returned from **requestSession()**.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, returns a **userInfo** referencing a **User** object and a 0 (zero) value. When it fails, this call returns a reference to an error text and a non-zero value.

Errors

106	User account does not exist
105	Internal error: SQL execution failed

See Also

The **User** class on page 16 and the **editUserInfo()** function on page 32.

editUserInfo()

Updates a user's non-phone attributes.

Syntax

```
function editUserInfo($enterpriseId,
                      $sessionId,
                      $UID
                      $userInfo,
                      &$errorText)
```

Description

editUserInfo() updates a user's non-phone attributes such as name, address, and gender. You pass into this function a **User** object with the updated information (**userInfo**). The object must contain at least a **firstName** and **lastName** value. Any attribute not set in **userInfo** variable remains unchanged in the existing **User** object.

Variables

enterpriseId	A IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
userInfo	A **User** object containing the attributes to update.

Returns

Upon success, a **User** object with the updated attribute and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **User** class on page 16.

addUserPhones()

Adds one or more phones to a system user.

Syntax

```
function addUserPhones($enterpriseId,
                       $sessionId,
                       $UID
                       $numbers,
                       &$errorText)
```

Description

addUserPhones() adds a phone number to the system user specified by the **UID** variable. The **numbers** variable takes an array of **PhoneNumber** objects.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
numbers	An array of class **PhoneNumber** objects to add.

Returns

On success, one or more **phoneId** values representing the newly added phone numbers and 0 (zero). Upon failure, this function returns the failure's cause in an **errorText** and a non-zero value.

Errors

105	Internal error: SQL execution failed
106	User account does not exist

See Also

The **PhoneNumber** class on page 19. The following functions **getPhoneId** on page 27, **removeUserPhones()** on page 34, on page 34, **setDefaultNumber()** on page 40.

removeUserPhones()

Removes one or more phone numbers from a user.

Syntax

```
function removeUserPhones($enterpriseId,
                          $sessionId,
                          $UID
                          $numbers,
                          &$errorText)
```

Description

removeUserPhones() removes one or more **PhoneNumber** objects from the system user with the specified **UID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
Numbers	An array of **PhoneNumber** objects to remove.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Error

105	Internal error: SQL execution failed
109	Cannot delete primary phone number

See Also

The **addUserPhones()** function on page 33.

getUserPhones()

Retrieves the phone numbers associated with a system user.

Syntax

```
function getUserPhones($enterpriseId,
                  $sessionId,
                  $UID
                  &$numbers,
                  &$errorText)
```

Description

getUserPhones() retrieves the an array of **PhoneNumber** objects associated with the user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, an array of **PhoneNumber** objects and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist

See Also

The **PhoneNumber** class on page 19. The following functions **addUserPhones()** on page 33, **removeUserPhones()** on page 34, and **setDefaultNumber()** on page 40.

setTryAllOption()

Sets the try-all attribute on a system user.

Syntax

```
function setTryAllOption($enterpriseId,
                         $sessionId,
                         $UID
                         &$errorText)
```

Description

setTryAllOption() sets the try all option on a system user. This function sets the try-all attribute on a system user. When try all is set, calls to this user cause the system to call all the phone number simultaneously in an attempt to contact the user. Calling the **unsetTryAllOption()** function turns this option off.

To override try-all attribute for specific blocks of time, call the **setSmsNumber()** function and set a **CallTimeException** on the user's preferences.

The system always first uses the user's call-time exception preferences. Then it considers the try-all function, calling all numbers simultaneously if this option is true. If try-all is false, the system attempts to contact the user using a hunt group. If the user has no hunt group set, the system uses the default number on the user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The function **unsetTryAllOption()** on page 37.

unsetTryAllOption()

Turns off the try-all attribute on a system user.

Syntax

```
function unsetTryAllOption($enterpriseId,
                           $sessionId,
                           $UID
                           &$errorText)
```

Description

unsetTryAllOption() turns off the try-all attribute on the system user specified by the **UID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **setTryAllOption()** on page 36.

setHuntOrder()

Sets a hunt order for a system user.

Syntax

```
function setHuntOrder($enterpriseId,
                      $sessionId,
                      $UID
                      $numbers,
                      $tryall,
                      &$errorText)
```

Description

setHuntOrder() sets the sequence of numbers that the system uses to call the user. Calling this function overwrites an existing hunt sequence.

Passing a value in the **tryall** variable sets or unsets this attribute on the user. If **tryall** is true, the system ignores the hunt order and rings all the phones simultaneously. If it is false, the system rings the numbers in the specified order until a person answers the call.

You can call this method with **numbers** that do not yet exist on the user. The system adds the number to the user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
Tryall	A Boolean value setting or unsetting the try-all attribute.
Numbers	An array of **PhoneNumber** objects arranged in the order of the hunt sequence.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **PhoneNumber** class on page 19. The **getHuntOrder()** function on page 39.

getHuntOrder()

Gets the user's current hunt order.

Syntax

```
function getHuntOrder($enterpriseId,
                      $sessionId,
                      $UID,
                      &$numbers,
                      &$tryall,
                      &$errorText)
```

Description

getHuntOrder() retrieves the hunt order for a user and returns it as an array of **PhoneNumber** objects. The objects return in the sequence order. The system also returns the current value of the user's try-all attribute.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, returns a **PhoneNumber** array in the hunt sequence order, the value of the try-all attribute, and a 0 (zero). Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **PhoneNumber** on page 19 and the **setHuntOrder()** function on page 38.

setDefaultNumber()

Sets the system user's default phone number.

Syntax

```
function setDefaultNumber($enterpriseId,
                          $sessionId,
                          $UID
                          $setPhoneNumber,
                          &$errorText)
```

Description

setDefaultNumber() sets the default **PhoneNumber** on the system for the specified **UID**. The system uses the default number to call a user as long as the user's preferences has no specified call exceptions, the **tryall** attribute is false, or no hunt order set.

If you specify a phone number that does not exist on the user, the system adds it before setting it as the default.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
number	A **PhoneNumber** object to set as the default.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist
110	Phone number not found

See Also

The **PhoneNumber** class on page 19 and **addUserPhones()** on page 33.

getDefaultNumber()

Gets the default number for a system user.

Syntax

```
function getDefaultNumber($enterpriseId,
                          $sessionId,
                          $UID
                          &$number,
                          &$errorText)
```

Description

getDefaultNumber() gets the default **PhoneNumber** object associated with the system user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, a reference to the phone **number** and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist
110	Phone number not found

See Also

The **setDefaultNumber()** function on page 40.

setSmsNumber()

Sets the system user's default phone number.

Syntax

```
function setSmsNumber($enterpriseId,
                      $sessionId,
                      $UID
                      $PhoneNumber,
                      &$errorText)
```

Description

setSmsNumber() sets a **PhoneNumber** as an SMS number for the specified **UID**. The system uses this number to send an SMS call to a user a user as long as the user's preferences has no specified call exceptions, the **tryall** attribute is false, or no hunt order set.

If you specify a phone number that does not exist on the user, the system adds it before setting it as an SMS number.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
number	A **PhoneNumber** object to set as an SMS number.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist
110	Phone number not found

See Also

The **PhoneNumber** class on page 19 and **addUserPhones()**on page 33.

getSmsNumber()

Gets an SMS number for a system user.

Syntax

```
function getSmsNumber($enterpriseId,
                      $sessionId,
                      $UID
                      &$number,
                      &$errorText)
```

Description

getSmsNumber() gets the default **PhoneNumber** object associated with the system user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, a reference to the phone **number** and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist
110	Phone number not found

See Also

The **setDefaultNumber()** function on page 40.

unsetSmsNumber()

Removes the SMS flag from a user account.

Syntax

```
function getSmsNumber($enterpriseId,
                      $sessionId,
                      $UID
                      &$PhoneNumber,
                      &$errorText)
```

Description

unsetSmsNumber() removes the SMS flag from the system user so that none of user's phones is set to receive SMS.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, returns a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist
110	Phone number not found

See Also

The **setDefaultNumber()** function on page 40.

setCallTimeException()

Sets a call time exception for a system user.

Syntax

```
function setCallTimeException($enterpriseId,
                              $sessionId,
                              $UID
                              $number,
                              $exceptionInfo,
                              &$errorText)
```

Description

setCallTimeException() sets a **CallTimeException** on the system user. A call time exception defines a number to call for a specified time period. A system user may have multiple **CallTimeException** objects. This exception supersedes default numbers and hunt sequences.

If the time period specified in this exception overlaps another already on the user, the new one will overwrite the previous exception where they overlap. For example, a system user may have a number preference from 9 AM to Noon and the new exception is from 11:30 to 5:00 PM. After you call this function, the user has two call time exceptions — one from 9 AM to 11:30 and another from 11:30 to 5:00 PM.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
number	The **PhoneNumber** number associated with this exception.
exceptionInfo	An array of **CallTimeException** objects to apply.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **`CallTimeException`** class on page 17. The **`removeCallTimeException()`** function on page 47.

removeCallTimeException()

Removes a call time exception from a system user.

Syntax

```
function setCallTimeException($enterpriseId,
                              $sessionId,
                              $UID
                              $number,
                              $exceptionInfo,
                              &$errorText)
```

Description

`setCallTimeException()` removes one or more **`CallTimeException`** objects from a user by overwriting the existing call time exceptions. You must explicitly declare the call times to remove. Meaning that if an exception is currently set for MTWTF for from 9-10 AM then you must set all 5 days in the **`CallTimeException`** object.

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`sessionId`	A session identifier returned from **`requestSession()`** function.
`UID`	The user's ID. The system assigns this value when you call **`createUser()`**.
`number`	The **`PhoneNumber`** object associated with the exception.
`exceptionInfo`	An array of **`CallTimeException`** objects to remove.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **`CallTimeException`** class on page 17.

getCallTimeExceptions()

Gets the call time exceptions associated a user.

Syntax

```
function getCallTimeException($enterpriseId,
                              $sessionId,
                              $UID
                              &$number,
                              &$exceptionInfo,
                              &$errorText)
```

Description

getCallTimeException() retrieves an array of **CallTimeException** objects on a system user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

On success, returns an array of **CallTimeException** objects, an array of **PhoneNumber** objects associated with them, and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **CallTimeException** class on page 17 and the **removeCallTimeException()** on page 47.

clearCallTimeException()

Clears all call time exceptions from a system user.

Syntax

```
function setCallTimeException($enterpriseId,
                              $sessionId,
                              $UID
                              &$errorText)
```

Description

clearCallTimeException() clears all **CallTimeException** objects from a user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

setDoNotDisturb()

Sets a do not disturb call time exception on a user.

Syntax

```
function setDoNotDisturb($enterpriseId,
            $sessionId,
            $UID,
            $exceptionInfo,
            &$errorText)
```

Description

setDoNotDisturb() sets a do-not-disturb (DND) **CallTimeException** on the system user. A DND call time exception defines a specified time period when the user will not accept phone calls. A system user may have multiple DND **CallTimeException** objects. This exception supersedes default numbers and hunt sequences.

If the time period specified in this exception overlaps another already on the user, the new one will overwrite the previous exception where they overlap. For example, a system user may have a DND preference from 9 AM to Noon and the new exception is from 11:30 to 5:00 PM. After you call this function, the user has two call time exceptions — one from 9 AM to 11:30 and another from 11:30 to 5:00 PM.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
exceptionInfo	An array of **CallTimeException** objects to apply.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

getDoNotDisturb()

Gets a user's DND call time exceptions.

Syntax

```
function getDoNotDisturb($enterpriseId,
          $sessionId,
          $UID,
          &$exceptionInfo,
          &$errorText)
```

Description

getDoNotDisturb() will return an array of **CallTimeException** objects currently assigned to a user. This array represents the user's do-not-disturb schedule.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

On success, returns an array of **CallTimeException** objects, an array of **PhoneNumber** objects associated with them, and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

removeDoNotDisturb()

Removes a do-not-disturb exception from a user.

Syntax

```
function removeDoNotDisturb($enterpriseId,
                            $sessionId,
                            $UID
                            $exceptionInfo,
                            &$errorText)
```

Description

removeDoNotDisturb() removes one or more **CallTimeException** objects from a user by overwriting the existing call time exceptions. You must explicitly declare the call times to remove. Meaning that if an exception is currently set for MTWTF for from 9-10 AM then you must set all 5 days in the **CallTimeException** object.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
exceptionInfo	An array of **CallTimeException** objects to remove.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

clearDoNotDisturb()

Clear all do-not-disturb exceptions from a user.

Syntax

```
function clearDoNotDisturb($enterpriseId,
          $sessionId,
          $UID,
          &$errorText)
```

Description

clearDoNotDisturb() clears all DND exceptions from a user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

resolveUserPhone()

Gets the preferred phone number from a user's preferences.

Syntax

```
function resolveUserPhone($enterpriseId,
                          $sessionId,
                          $UID,
                          $time,
                          &$number,
                          &tryall,
                          &$errorText)
```

Description

resolveUserPhone() given a specific time, this function gets the preferred phone number from user preferences. User preferences contain **CallTimeException** objects that define where a user prefers to receive a call at a specific time. You must specify both a **time** and a **UID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
time	The time to resolve specified in UNIX GMT time.

Returns

Upon success, returns a reference to the **PhoneNumber**, the **tryall** attribute value and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

106	User account does not exist

See Also

The classes **CallTimeException** on page 3 and **PhoneNumber** on page 19.

blackListPhone()

Blocks a telephone number from calling a user.

Syntax

```
function blackListPhone($enterpriseId,
                  $sessionId,
                  $UID,
                  $numbers,
                  &$errorText)
```

Description

blackListPhone() prevents one or more **PhoneNumber** objects from calling a specific user. Use the **removePhoneFromBlackList()** function to remove the phone number from the list.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
numbers	An array of **PhoneNumber** objects to block.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
113	Duplicate phone number

See Also

The functions **getBlackList()** on page 56 and **removePhoneFromBlackList()** on page 57.

getBlackList()

Returns a users blacklisted phone numbers.

Syntax

```
function getBlackList($enterpriseId,
                      $sessionId,
                      $UID,
                      &$numbers,
                      &$errorText)
```

Description

getBlackList() returns a user's blacklist as an array of **PhoneNumber** objects.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, returns an array of black listed **PhoneNumber** objects and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **blackListPhone()** on page 55 and the **removePhoneFromBlackList()** on page 57.

removePhoneFromBlackList()

Removes a number from a blacklist.

Syntax

```
function removePhoneFromBlackList($enterpriseId,
                                  $sessionId,
                                  $UID,
                                  $numbers,
                                  &$errorText)
```

Description

removePhoneFromBlackList() removes a number previously added to a user's blacklisted numbers. Once removed, the number can call to the user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
numbers	An array **PhoneNumber** objects to remove from the user's blacklist.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **blackListPhone()** on page 55 **getBlackList()** on page 56.

blockUsers()

Blocks one or more users from calling a user.

Syntax

```
function blockUsers($enterpriseId,
                    $sessionId,
                    $UID,
                    $users,
                    &$errorText)
```

Description

blockUsers() prevents one or more users from calling a specific user. This blocks all of the numbers associated with one or more users from calling the user specified by **UID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
user	An array of **User** objects to block.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

unblockUser()

Removes one or more blocks.

Syntax

```
function unblockUser($enterpriseId,
                     $sessionId,
                     $UID,
                     $users,
                     &$errorText)
```

Description

unblockUsers() removes the block that prevents one user from calling another (**UID**).

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
user	An array of **User** objects to unblock.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

getBlockedUsers()

Blocks one or more users from calling a user.

Syntax

```
function getBlockedUsers($enterpriseId,
                  $sessionId,
                  $UID,
                  &$blockedUsers,
                  &$errorText)
```

Description

getBlockedUsers() returns an array of users that the specified user (**UID**) is blocking.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

Upon success, this function returns 0 (zero) and an array of user objects. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

addUserEmail()

Associates one or more email addresses with a user.

Syntax

```
function addUserEmail($enterpriseId,
                      $sessionId,
                      $UID
                      $emailAddresses,
                      &$errorText)
```

Description

addUserEmail() adds an array of email addresses to a user's preferences. Each address in **emailAddresses** must be unique. The system ensures the addresses resolve.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
emailAddresses	An array of email addresses to assign.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist

See Also

The functions **getUserEmails()** on page 62, **removeUserEmail()** on page 63, **setDefaultEmail()** on page 64 and **getUIDbyEmail()** on page 65.

getUserEmails()

Gets the emails associated with a user.

Syntax

```
function getUserEmails($enterpriseId,
                       $sessionId,
                       $UID
                       &$defaultEmail,
                       &$emailAddresses,
                       &$errorText)
```

Description

getUserEmails() returns the user's single default email and an array of all the other email addresses in the user's preferences.

Variables

enterpriseId	An IntelePeer enterprise ID. This value is required.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

On success, returns the user's default email address, an array of all the user's **emailAddresses**, and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist

See Also

removeUserEmail() on page 63.

removeUserEmail()

Removes one or more user email addresses.

Syntax

```
function removeUserEmail($enterpriseId,
                         $sessionId,
                         $UID
                         $emailAddresses,
                         &$errorText)
```

Description

removeUserEmail() given an array of **emailAddresses**, removes the addresses from the associated user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
emailAddresses	An array of email addresses to remove.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
107	Incorrect email address for user
108	Cannot delete primary email address

See Also

getUserEmails() on page 62.

setDefaultEmail()

Sets the user's default email address.

Syntax

```
function setDefaultEmail($enterpriseId,
                  $sessionId,
                  $UID,
                  $emailAddress,
                  &$errorText)
```

Description

setDefaultEmail() sets the default email address in a user's preferences. The user's default email address must be unique.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
emailAddress	The email address to set as the default.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
106	User account does not exist

See Also

The **getUserEmails()** on page 62 and **removeUserEmail()** on page 63.

getUIDbyEmail()

Gets the user ID associated with an email address.

Syntax

```
function getUIDbyEmail($enterpriseId,
                       $sessionId,
                       $userEmail,
                       &$UID,
                       &$errorText)
```

Description

getUIDbyEmail() given a **userEmail** this function returns the **UID** associated with it.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
userEmail	A email to user to locate a user.

Returns

On success, a reference to **UID** and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

getCarriers()

Gets the carriers associated with a user.

Syntax

```
function getCarriers($enterpriseId,
                     $sessionId,
                     $UID,
                     &$carriers,
                     &$errorText)
```

Description

getCarriers() returns the **carriers** associated with a user (**UID**) through its associated phone numbers. For each carrier, the system returns the carrier ID, name, and gateway.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

On success, an array of **carriers** on a user and a 0 (zero). On failure, the **errorText** specifying the cause and a non-zero return value.

Errors

105	Internal error: SQL execution failed
111	Internal error: System data missing or corrupt

See Also

The **PhoneNumber** class on page 19.

setUserCurrency()

Sets the user's currency.

Syntax

```
function setUserCurrency($enterpriseId,
                         $sessionId,
                         $UID,
                         $amount,
                         &$errorText)
```

Description

setUserCurrency() sets the currency in the user's preferences to the specified **amount**. A user can have a negative value for currency. The system allows you to specify a negative integer for the **amount**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
Amount	The amount to apply to the user.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **getUserCurrency()** on page 68, **incrementUserCurrency()** on page 69, and **decrementUserCurrency()** on page 70.

getUserCurrency()

Gets the amount of a user's currency.

Syntax

```
function getUserCurrency($enterpriseId,
                         $sessionId,
                         $UID,
                         &$amount,
                         &$errorText)
```

Description

getUserCurrency() gets the current amount of the user's currency.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

On success, returns the **amount** currently available to the user and 0 (zero). Upon failure, this function returns the failure's cause and false.

Errors

105	Internal error: SQL execution failed

See Also

The functions **setUserCurrency()** on page 67, **incrementUserCurrency()** on page 69, and **decrementUserCurrency()** on page 70.

incrementUserCurrency()

Increases the amount of a user's currency.

Syntax

```
function incrementUserCurrency($enterpriseId,
                               $sessionId,
                               $UID,
                               $amount,
                               &$errorText)
```

Description

incrementUserCurrency() increments the user's currency by the specified **amount**. The system allows you to specify a negative integer for the **amount**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
Amount	The amount by which to increment the user currency.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **getCarriers()** on page 66, **getUserCurrency()** on page 68, and **decrementUserCurrency()** on page 70.

decrementUserCurrency()

Reduces the amount of a user's currency.

Syntax

```
function decrementUserCurrency($enterpriseId,
                               $sessionId,
                               $UID,
                               $amount,
                               &$errorText)
```

Description

decrementUserCurrency() subtracts the specified **amount** from the user's currency. A user can have a negative currency value.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
Amount	The amount to subtract.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **getUserCurrency()** on page 68, and **incrementUserCurrency()** on page 69.

setMOUAllowed()

Sets the minutes of use (MOU) for a user.

Syntax

```
function setMOUAllowed($enterpriseId,
                       $sessionId,
                       $UID,
                       $mou,
                       &$errorText)
```

Description

setMOUAllowed() allocates the specified **mou** to the specified **UID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
Mou	The minutes to allocate.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **resetMOUused()** function on page 74.

getMOUAllowed()

Gets the minutes of use (MOU) for a user.

Syntax

```
function getMOUAllowed($enterpriseId,
                       $sessionId,
                       $UID,
                       &$mou,
                       &$errorText)
```

Description

getMOUAllowed() gets the user's allocated MOU.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The **resetMOUused()** function on page 74.

getMOUused()

Gets the number of minutes used.

Syntax

```
function getMOUused($enterpriseId,
                    $sessionId,
                    $UID,
                    &$mou,
                    &$errorText)
```

Description

getMOUused() given a specific **UID** this function returns the accumulated **mou** value.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

On success, returns a user's accumulated **mou** and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functison **setMOUAllowed()** on page 71 and **resetMOUused()** on page 74.

resetMOUused()

Sets a user's MOU to 0 (zero).

Syntax

```
function resetMOUused($enterpriseId,
                      $sessionId,
                      $UID,
                      &$errorText)
```

Description

resetMOUused() sets the MOU for the specified **UID** to 0 (zero).

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **setMOUAllowed()** on page 71.

createGroup()

Creates a call group.

Syntax

```
function createGroup($enterpriseId,
                     $sessionId,
                     $UID,
                     $groupName,
                     &$GID,
                     &$errorText)
```

Description

createGroup() creates a call group for the specified **UID**. For example, a group called Family would hold all of a user's family numbers. You first create a group and then **addToGroup()** to place a set of **EndpointIdentifiers** in the group. In this way, a group can contain contacts that reference both system users and non-users.

☞ If an **EndpointIdentifier** is of ID_TYPE_PHONE, the system never matches that phone number to a user record even if it belongs to an existing system user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
groupName	The group's display name.

Returns

On success, returns the **GID** assigned to the group and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **renameGroup()** on page 76, **deleteGroup()** on page 77, **deleteFromGroup()** on page 78, **addToGroup()** on page 79, and **getGroupNumbers()** on page 80.

renameGroup()

Renames a group.

Syntax

```
function renameGroup($enterpriseId,
                     $sessionId,
                     $UID,
                     $GID,
                     $groupName,
                     &$errorText)
```

Description

renameGroup() renames the group with the specified GID on the user preferences.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
GID	The group's ID. The system assigns this value when you create the group.
groupName	The group's new display name.

Returns

On success, returns a 0. Upon failure, this function returns an array containing a non-zero error code and an error message.

Error

105	Internal error: SQL execution failed

See Also

The function **createGroup()** on page 75, **deleteGroup()** on page 77, **deleteFromGroup()** on page 78, **addToGroup()** on page 79, and **getGroupNumbers()** on page 80.

deleteGroup()

Removes a group from user preferences.

Syntax

```
function deleteGroup($enterpriseId,
                     $sessionId,
                     $UID,
                     $GID,
                     &$errorText)
```

Description

deleteGroup() deletes the group with the specified **GID** from the user preferences. This removes the group only. Any users referenced by the group's **EndpointIdentifiers** remain in the system; you are only deleting organizational structure.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
GID	The group's ID. The system assigns this value when you create the group.

Returns

On success, returns a 0. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
112	Group not owned by user

See Also

The functions **createGroup()** on page 75, **renameGroup()** on page 76, **deleteFromGroup()** on page 78, **addToGroup()** on page 79, and **getGroupNumbers()** on page 80.

deleteFromGroup()

Deletes one or more numbers from a group.

Syntax

```
function deleteFromGroup($enterpriseId,
                         $sessionId,
                         $UID,
                         $GID,
                         $memberIds,
                         &$errorText)
```

Description

deleteFromGroup() takes an array of **memberIds** and removes the association between the phone numbers they reference and the group.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
GID	The group's ID. The system assigns this value when you create the group.
memberIds	An array of **EndpointIdentifier** to remove from the group.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed

See Also

The functions **createGroup()** on page 75, **renameGroup()** on page 76, **deleteGroup()** on page 77, **addToGroup()** on page 79, and **getGroupNumbers()** on page 80.

addToGroup()

Adds additional numbers to an existing group.

Syntax

```
function addToGroup($enterpriseId,
                    $sessionId,
                    $UID,
                    $GID,
                    $identifiers,
                    &$errorText)
```

Description

addToGroup() takes an array of **identifiers** objects and adds them the user's group identified by the **GID**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
GID	The group's ID. The system assigns this value when you create the group.
Indentifiers	An array of **EndpointIdentifier** objects to add to the group

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
115	Group not found

See Also

The **EndpointIdentifier** class on page 3. The functions **createGroup()** on page 75, **renameGroup()** on page 76, **deleteGroup()** on page 77, **deleteFromGroup()** on page 78, and **getGroupNumbers()** on page 80.

getGroupNumbers()

Gets the numbers in a group.

Syntax

```
function getGroupNumbers($enterpriseId,
                    $sessionId,
                    $UID,
                    $GID,
                    &$members,
                    &$errorText)
```

Description

getGroupNumbers() Queries the group on the user specified by **UID** and returns its **members** as an array of **EndpointIdentifier** objects.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
GID	The group's ID. The system assigns this value when you create the group.

Returns

On success, returns an array of **EndpoinIdentifier** objects representing the group members and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

105	Internal error: SQL execution failed
115	Group not found

See Also

The functions **createGroup()** on page 75, **renameGroup()** on page 76, **deleteGroup()** on page 77, **deleteFromGroup()** on page 78, **addToGroup()** on page 79.

clickToCall()

Makes an immediate call.

Syntax

```
function clickToCall($enterpriseId,
                $sessionId,
                $UID
                $originationId,
                $terminationId,
                $hideAni,
                $announcementText,
                $disableAMD,
                &$callIdentifier,
                &$errorText)
```

Description

clickToCall() creates a two-way call bridge between the two **EndpointIdentifier** objects specified by the **originationId** and **terminationId**. As with all OCP calls, the system first connects to the termination side.

The structure of the termination **EndpointIdentifier** determines how the system reaches the termination side. The type of call, in this case click-to-call, determines what the system accepts as a successful answer. By default, **clickToCall()** function connects the termination side when it detects a human response. Answering machine responses are failures.

Upon answer, the system plays an announcement to the call recipient instructing the recipient to press 1 to accept. When the recipient accepts the call, the system calls the **originationId** point and bridges the two call legs.

By default, the system attempts to detect an answering machine on the termination side. This means, when the system connects it waits to hear a human voice or an answering machine respond. If you are testing the system and you receive a call, you must respond to activate the call sequence. If you wish the system to ignore detection and go straight to the announcement, set **disableAMD**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
originationId	An **EndpointIdentifier** that describes the call's origination side.
terminationIds	An **EndpointIdentifier** that describes the call's termination side.

hideAni	A boolean value that determines whether or not the system presents the caller's number to the recipient's caller ID.
announcementText	The announcement text to play to the call termination point.
disableAMD	Sets answering machine detection on or off.

Returns

On success, the system returns a unique call ID and a 0 (zero) value. IntelePeer Services assigns the ID which you can use to reference the call in subsequent API calls. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
302	Invalid Origination Type supplied
303	Blacklisted origination
304	Internal error: SQL execution failed

clickToCallOrig()

Makes an immediate call.

Syntax

```
function clickToCallOrig($enterpriseId,
                $sessionId,
                $UID
                $originationId,
                $terminationId,
                $hideAni,
                $announcementText,
                $disableAMD,
                &$callIdentifier,
                &$errorText)
```

Description

clickToCallOrig() creates a two-way call bridge between the two **EndpointIdentifier** objects specified by the **originationId** and **terminationId**.

Unlike other OCP calls, the system first connects a **clickToCallOrig()** call to the origination side first. This means the system will ignore constraints on the termination side and connect the call regardless of whether an answering machine picks up or not. The benefit of this feature is that the call originator can leave a message on an answering machine. To turn this behavior off, set the **callOrigination** value to 0 (false).

Upon answer, the system plays an announcement to the call recipient instructing the recipient to press 1 to accept. When the recipient accepts the call, the system calls the **terminationId** point and bridges the two call legs.

By default, the system attempts to detect an answering machine on the termination side. This means, when the system connects it waits to hear a human voice or an answering machine respond. If you are testing the system and you receive a call, you must respond to activate the call sequence. If you wish the system to ignore detection and go straight to the announcement, set **disableAMD** to true.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
callOrigination	When 1 (true), the system connects the origination side before the termination side. This is the default. Set the value to 0 (false) to turn this off.

originationId	An **EndpointIdentifier** that describes the call's origination side.
terminationIds	An **EndpointIdentifier** that describes the call's termination side.
hideAni	A boolean value that determines whether or not the system presents the caller's number to the recipient's caller ID.
announcementText	The announcement text to play to the call termination point.
disableAMD	Sets answering machine detection on or off.

Returns

On success, the system returns a unique call ID and a 0 (zero) value. IntelePeer Services assigns the ID which you can use to reference the call in subsequent API calls. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
302	Invalid Origination Type supplied
303	Blacklisted origination
304	Internal error: SQL execution failed

scheduleAutoConference()

Dials a user into a conference.

Syntax

```
function scheduleAutoConference($enterpriseId,
                                $sessionId,
                                $UID,
                                $conferenceNumber,
                                $conferencePin,
                                $originationId,
                                $conferenceTime,
                                $announcementText,
                                $disableAMD,
                                &$callIdentifier,
                                &$errorText)
```

Description

scheduleAutoConference() dials and connects a user (**UID**) into a conference at the specified **conferenceTime**. The system connects to the user first using the user's call preferences. The **scheduleAutoConference()** function considers the connection to the user a success when it reaches a human voice, it ignores answering machines or voice mail.

Upon answering the call, the user does not need the conference number or pin. Instead, the system presents an announcement to the user. The user presses 1 to accept the call and the system joins the user to the conference. At this point, the system initiates the second leg of the call to the conference number. The system supplies the **conferencePin** to the conference engine and then bridges the conference (origination) and the user (termination) legs of the call.

By default, the system attempts to detect an answering machine on the termination side. This means, when the system connects it waits to hear a human voice or an answering machine respond. If you are testing the system and you receive a call, you must respond to activate the call sequence. If you wish the system to ignore detection and go straight to the announcement, set **disableAMD**.

The system applies the MOU for this conference to the user.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The conference requester's user ID. The system assigns the ID when you create the user.
conferenceNumber	The conference telephone number.

conferencePin	The conference's PIN number. The system outpulses these digits as DTMF (touch tones) to the conference engine. Valid digits are 0-9, # (pound sign), and * (asterisk).
originationId	An **EndpointIdentifier** describing the identity of the origination side of the call.
conferenceTime	The time to contact the conference participant. Specify this value as UNIX time in GMT. Set to 0 (zero) for no expiration.
announcementText	The announcement text to play to the call termination point.
disableAMD	Sets answering machine detection on or off.

Errors

301	Invalid Termination Type supplied
302	Invalid Origination Type supplied
304	Internal error: SQL execution failed

Returns

On success, the system returns a unique call ID and a 0 (zero) value. IntelePeer Services assigns the **callIdentifier** which you can use to reference the call in subsequent API calls. Upon failure, this function returns an array containing a non-zero error code and an error message.

getAutoConferenceDetails()

Gets an auto conference's parameters.

Syntax

```
function getAutoConferenceDetails($enterpriseId,
                                  $sessionId,
                                  $UID
                                  $callIdentifier,
                                  &$conferenceParams,
                                  &$errorText)
```

Description

getAutoConferenceDetails() retrieves the set of conference parameters associated with an auto conference and a particular **UID**. The **UID** identifies the user who joined the conference. You must specify the IntelePeer assigned **callIdentifier**. The function supplies the following parameters in an associative array:

- **conferenceNumber**
- **conferencePin**
- **originationId**
- **conferenceTime**
- **announcementText**

You can use the associative array populated by **getAutoConferenceDetails()** in calls to **Errors**

304	Internal error: SQL execution failed
305	Conference does not exist
307	Internal error: Corrupted Data

editAutoConference().

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The conference requester's user ID. The system assigns the ID when you create the user.
callIdentifier	The **callIdentifier** returned from the **clickToCallOrig()** function.

Returns

On success, an associative array in a **`conferenceParams`** value and 0. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
305	Conference does not exist
307	Internal error: Corrupted Data

editAutoConference()

Edits auto conference parameters of a scheduled conference.

Syntax

```
function editAutoConferenceDetails($enterpriseId,
                                   $sessionId,
                                   $UID,
                                   $callIdentifier,
                                   $conferenceParams
                                   &$errorText)
```

Description

editAutoConferenceDetails() updates the parameters of a conference scheduled with a **clickToCallOrig()** function. You supply to this call the **callIdentifier** returned when you scheduled the call. You also supply the **UID** of the user who requested the conference. You must also specify a set of **conferenceParams** as an associate array with the following parameters:

- **conferenceNumber**
- **conferencePin**
- **originationId**
- **conferenceTime**
- **announcementText**

Format for the values are the same as in the **scheduleAutoConference()** call. If you omit parameters in the array, the system leaves the original values unchanged.

Variables

EnterpriseId	An IntelePeer enterprise ID..
SessionId	A session identifier returned from **requestSession()** function.
UID	The conference requester's user ID. The system assigns the ID when you create the user.
CallIdentifier	The **callIdentifier** returned from the **clickToCallOrig()** call.
ConferenceParams	An associative array containing the parameters you wish to change.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
302	Invalid Origination Type supplied
304	Internal error: SQL execution failed

getAutoConferenceList()

Returns an array of callIds for a scheduled conference.

Syntax

```
function editAutoConferenceDetails($enterpriseId,
                                   $sessionId,
                                   $UID,
                                   &callIds,
                                   &$errorText)
```

Description

`getAutoConferenceList()` returns an array of call identifiers for a scheduled autoconference. The system creates these call identifiers when you add legs to a conference.

Variables

`EnterpriseId`	An IntelePeer enterprise ID..
`SessionId`	A session identifier returned from **`requestSession()`** function.
`UID`	The conference requester's user ID. The system assigns the ID when you create the user.

Returns

Upon success, an array of call identifiers and a 0 (zero). Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
305	Conference does not exist
307	Internal error: Corrupted Data

deleteAutoConference()

Removes a scheduled auto conference.

Syntax

```
function deleteAutoConference($enterpriseId,
                              $sessionId,
                              $UID,
                              $callIdentifier,
                              &$errorText)
```

Description

Use this function to delete a conference you have scheduled with a **clickToCallOrig()** call. You supply to this call the **callIdentifier** returned when you scheduled the call and the **UID** of the user who requested the conference.

Variables

EnterpriseId	An IntelePeer enterprise ID.
UID	The conference requester's user ID. The system assigns the ID when you create the user.
SessionId	A session identifier returned from **requestSession()** function.
callIdentifier	The **callIdentifier** returned from the **clickToCallOrig()** call..

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed

See Also

getAutoConferenceDetails() on page 87

conferenceOnDemand()

Creates a conference on demand call.

Syntax

```
function conferenceOnDemand($enterpriseId,
                        $sessionId,
                        $UID,
                        $terminationID,
                        $conferenceDuration,
                        $announcementText,
                        $disableAMD,
                        &$conferenceId,
                        &$callIdentifier,
                        &$errorText)
```

Description

conferenceOnDemand() creates a conference call on behalf of the specified moderator identified by the **UID**. This function establishes the virtual "conference room" and places the moderator into it. The conference is initiated the moment this call is successfully made. Your application must use **addLegToConference()** to add participants.

You supply an **EndpointIdentifier** object in the **terminationId** parameter. This termination point represents the conference moderator. The system attempts to contact the moderator according to the structure of the **EndpointIdentifier**. The system considers it a success when it reaches a human voice. At that point, the system plays the **announcementText** and prompts the user to press 1 to accept the call and join the conference. The participant need not know the **conferenceId** or PIN.

The **conferenceDuration** is the amount of time the moderator wishes to reserve for the conference. The conference may go exceed its duration however you should ensure that you call to the **extendConference()** function to reserve the conference room's time.

By default, the system attempts to detect an answering machine on the termination side. This means, when the system connects it waits to hear a human voice or an answering machine respond. If you are testing the system and you receive a call, you must respond to activate the call sequence. If you wish the system to ignore detection and go straight to the announcement, set **disableAMD**.

The system applies the conference's duration to the moderator's MOU.

Variables

enterpriseId	An IntelePeer enterprise ID..
sessionId	A session identifier returned from **requestSession()** function.
UID	The moderator's user ID. The system assigns this value when you call **createUser()**.

terminationId	An **EndpointIdentifier** describing the identity of the origination side of the call. This is the moderator.
conferenceDuration	A UNIX time value representing the duration in time that the conference is expected to last.
announcementText	The announcement text to play to the conference participants.
disableAMD	Sets answering machine detection on or off.

Returns

On success, the system returns unique **conferenceId** and the moderator's **callIdentifier** value. IntelePeer Services assigns these IDs which you can use to reference the call in subsequent API calls. The system returns a callIdentifier for each termination point.

Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
303	Blacklisted origination
304	Internal error: SQL execution failed

addLegToConference()

Adds one or more user to an existing conference on demand.

Syntax

```
function addLegToConference($enterpriseId,
                        $sessionId,
                        $UID,
                        $conferenceId,
                        $terminationIds,
                        $announcementText,
                        &$callIdentifier,
                        &$errorText)
```

Description

addLegToConference() adds one or more users to the conference on demand with the specified **conferenceId**. Pass an array of **terminationIds** containing **EndpointIdentifier** objects representing the participants to add. You must specify which conference using the **conferenceId** value.

The system calls each user and plays the **announcementText**. After pressing 1 to accept the conference, the user is added into the conference. The system returns an associative array of **callIdentifier** objects, one for each **EndpointIdentifier** specified.

Variables

enterpriseId	An IntelePeer enterprise ID..
sessionId	A session identifier returned from **requestSession()** function.
UID	The moderator's user ID. The system assigns this value when you call **createUser()**.
conferenceId	The unique ID assigned to the conference by the **conferenceOnDemand()** function.
terminationId	The participants to add. This can be a single participant or a group. If the value represents a group, the system adds all the members of the group to the conference.
announcementText	The announcement text to play to the call participants.

Returns

On success, an array of IntelePeer generated **callIdentifier** objects and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
303	Blacklisted origination
304	Internal error: SQL execution failed

See Also

The functions **`addLegToConference()`** on page 95 and **`extendConference()`** on page 97.

extendConference()

Extends the duration of a conference on demand.

Syntax

```
function extendConference($enterpriseId,
                          $sessionId,
                          $UID,
                          $conferenceId,
                          $extensionDuration
                          &$errorText)
```

Description

`extendConference()` extends the time a conference bridge is reserved for a **`conferenceOnDemand`**. If you do not extend a conference and the conference goes beyond the specified duration, the system will not reserve the conference. This means that the next conference in the system queue can take your conference room.

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`sessionId`	A session identifier returned from **`requestSession()`** function.
`UID`	The moderator's user ID. The system assigns this value when you call **`createUser()`**.
`conferenceId`	The ID of the conference returned by the **`conferenceOnDemand()`** call
`extensionDuration`	A value in UNIX time representing the amount of time to extend the conference.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

305	Conference does not exist
306	Conference cannot not be extended

See Also

The functions **`conferenceOnDemand()`** on page 93 and **`addLegToConference()`** on page 95.

getCallIdsByConfID()

Gets the call Ids associated with a conference on demand call.

Syntax

```
function getCallIdsByConfID($enterpriseId,
               $sessionId,
               $confId,
               $UID,
               &$callIdentifiers,
               &$errorText)
```

Description

getCallIdsByConfID() retrieves the IDs of a completed conference-on-demand and returns them in an array of **callIdentifiers**.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
confID	The ID assigned by IntelePeer and returned by the conference creation function.

Returns

On success, an array of **callIdentifiers** and a 0 (zero) value. Each **callIdentifier** includes the following information:

- **callId**
- **mou**
- **number**
- **reason**
- **status**
- **timeScheduled**
- **timeOriginated**
- **timeCompleted**
- **callingNumber**
- **currentState**
- **callComplete**

Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

3	Required parameters missing

blastVoice()

Blasts a message to multiple recipients.

Syntax

```
function blastVoice($enterpriseId,
                $sessionId,
                $UID,
                $originationId,
                $terminationIds,
                $audioFile,
                $blastTime,
                $blastTTS,
                $audioFile,
                $hideAni=false,
                $allowCallback=true,
                &$blastId,
                &$errorText)
```

Description

blastVoice() sends a message to one or more recipients at the specified **blastTime**. Typically, this function is used for reminders. You specify the recipients by passing in an array of **terminationIds**. The UID specifies which user originates the blast.

You can send a text blast or an audio blast. The system converts the text specified by **blastTTS** to audio and presents it to the termination side of the call. If you specify an audio blast, you supply a File object in the **audioFile** parameter. This object represents an or an audio clip on the Internet via a **url**.

The **originationId** points to the **EndpointIdentifier** for the blast originator. When **allowCallback** is true, recipients may have the option to call back to the originator by pressing 1. By default, **blastVoice()** presents the blast originator's number to blast recipients. Set the **hideAni** value to true to hide the originator's number.

To blast to user groups use the **resolveGroup()** function.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The ID of the user initiating the blast. The system generates this value when you create the user.
terminationIds	An array of **EndpointIdentifier** objects.

blastTime	The time in GMT when the blast is schedule to happen.
blastTTS	A text string to use in the blast.
audioFile	A **File** object representing an MP3 or WMA file to play to the recipient.
hideAni	A Boolean value specifying whether the caller's number from the recipient's caller ID. By default, this false and the system reveals the number.
allowCallback	A Boolean value specifying if the recipient can call back to the blast originator. By default, this is true and the system allows the recipient to call back.

Returns

On success, an IntelePeer generated identifier for the blast call and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
302	Invalid Origination Type supplied
304	Internal error: SQL execution failed

getBlastVoice()

Gets the details associated with a scheduled blast.

Syntax

```
function getBlastVoice($enterpriseId,
                  $sessionId,
                  $UID,
                  $blastId,
                  &$blastDetails,
                  &$errorText)
```

Description

getBlastVoice() retrieves the details for the scheduled blast identified by the **blastId** parameter. This function returns an array containing the values for the following:

- **allowCallback**
- **announcementText**
- **blastTime**
- **hideAni**
- **originationId**
- **terminationIds**
- **audioFile**

You set these values originally when you created the blast with the **blastVoice()** function.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
blastId	The blast ID assigned by IntelePeer.

Returns

On success, an array of blast details and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
307	Internal error: Corrupted Data

308 Blast not found

See Also

The functions **`blastVoice()`**on page 101 and **`deleteBlastVoice()`** on page 119.

editBlastVoice()

Updates a blast reminder.

Syntax

```
function editBlastVoice($enterpriseId,
         $sessionId,
         $UID,
         $blastId,
         $originationId,
         $terminationIds,
         $blastTime,
         $blastTTS,
         $audioFile,
         $hideAni=false,
         $allowCallback=true,
         &$errorText)
```

Description

editBlastVoice() takes a **blastId** that corresponds to a scheduled blast and edits the information associated with the scheduled blast. The **originationId** points to the **EndpointIdentifier** for the blast originator.

When **allowCallback** is true, recipients may have the option to call back to the originator by pressing 1. By default, **blastVoice()**presents the blast originator's number to blast recipients. Set the **hideAni** value to true to hide the originator's number.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The ID of the user initiating the blast. The system generates this value when you create the user.
blastId	The identifier of the blast call you want to edit.
originationId	An **EndpointIdentifier** object specifying the origination point.
terminationIds	An array of **EndpointIdentifier** objects.
blastTime	The time in GMT when the blast is schedule to happen.
blastTTS	A text string to use in the blast.

audioFile	A **File** object representing an MP3 or WMA file to play to the recipient.
hideAni	A Boolean value specifying whether the caller's number from the recipient's caller ID. By default, this false and the system reveals the number.
allowCallback	A Boolean value specifying if the recipient can call back to the blast originator. By default, this is true and the system allows the recipient to call back.

Returns

On success, a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
307	Internal error: Corrupted Data
308	Blast not found

See Also

The functions **blastVoice()**on page 101 and **deleteBlastVoice()** on page 119.

getBlastVoiceList()

Returns an array of blast identifiers for a scheduled blast voice event.

Syntax

```
function editAutoConferenceDetails($enterpriseId,
                                   $sessionId,
                                   $UID,
                                   &blastIds,
                                   &$errorText)
```

Description

`getBlastVoiceList()` returns an array of blast identifiers for a scheduled blast voice events. The system creates these call identifiers when you add recipients to a blast voice event.

Variables

`EnterpriseId`	An IntelePeer enterprise ID..
`SessionId`	A session identifier returned from **`requestSession()`** function.
`UID`	The conference requester's user ID. The system assigns the ID when you create the user.

Returns

Upon success, an array of blast identifiers and a 0 (zero). Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
308	Blast not found

blastSms()

Blasts an SMS text message to multiple recipients.

Syntax

```
function blastSms($enterpriseId,
                  $sessionId,
                  $UID,
                  $originationId,
                  $terminationIds,
                  $blastTime,
                  $blastTTS,
                  &$blastId,
                  &$errorText)
```

Description

`blastSms()` sends an SMS message to one or more recipients at the specified **`blastTime`**. Typically, this function is used for reminders. You specify the recipients by passing in an array of **`terminationIds`**. The UID specifies which user originates the blast.

The **`originationId`** points to the **`EndpointIdentifier`** for the blast originator. To blast to user groups use the **`resolveGroup()`** function.

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`sessionId`	A session identifier returned from **`requestSession()`** function.
`UID`	The ID of the user initiating the blast. The system generates this value when you create the user.
`terminationIds`	An array of **`EndpointIdentifier`** objects.
`blastTime`	The time in GMT when the blast is schedule to happen.
`blastTTS`	A text string to use in the blast.

Returns

On success, an IntelePeer generated identifier for the blast call and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

301	Invalid Termination Type supplied
302	Invalid Origination Type supplied
304	Internal error: SQL execution failed

getBlastSms()

Gets the details associated with a scheduled blast.

Syntax

```
function getBlastSms($enterpriseId,
                     $sessionId,
                     $UID,
                     $blastId,
                     &$blastDetails,
                     &$errorText)
```

Description

getBlastSms() retrieves the details for the scheduled SMS blast identified by the **blastId** parameter. This function returns an array containing the values for the following:

- **announcementText**
- **blastTime**
- **originationId**
- **terminationIds**

You set these values originally when you created the blast with the **blastSms()** function.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
blastId	The blast ID assigned by IntelePeer.

Returns

On success, an array of blast details and a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
307	Internal error: Corrupted Data
308	Blast not found

See Also

The functions **`blastVoice()`**on page 101 and **`deleteBlastVoice()`** on page 119.

editBlastSms()

Updates an SMS blast reminder.

Syntax

```
function editBlastSms($enterpriseId,
         $sessionId,
         $UID,
         $blastId,
         $originationId,
         $terminationIds,
         $blastTime,
         $blastTTS,
         &$errorText)
```

Description

editBlastSms() takes a **blastId** that corresponds to a scheduled blast and edits the information associated with the scheduled blast. The **originationId** points to the **EndpointIdentifier** for the blast originator.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The ID of the user initiating the blast. The system generates this value when you create the user.
blastId	The identifier of the blast call you want to edit.
originationId	An **EndpointIdentifier** object specifying the origination point.
terminationIds	An array of **EndpointIdentifier** objects.
blastTime	The time in GMT when the blast is schedule to happen.
blastTTS	A text string to use in the blast.

Returns

On success, a 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
307	Internal error: Corrupted Data
308	Blast not found

See Also

The functions **`blastVoice()`**on page 101 and **`deleteBlastVoice()`** on page 119.

deleteBlastVoiceSms()

Deletes a scheduled blast.

Syntax

```
function deleteBlastVoiceSms($enterpriseId,
                  $sessionId,
                  $UID,
                  $blastId,
                  &$errorText)
```

Description

deleteBlastVoiceSms() deletes a scheduled SMS blast identified by the **blastId** parameter.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
blastId	The blast ID assigned by IntelePeer.

Returns

On success, 0 (zero) value. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
307	Internal error: Corrupted Data
308	Blast not found

See Also

The functions **blastVoice()**on page 101 and **deleteBlastVoice()** on page 119.

getBlastSmsList()

Returns an array of blast identifiers for a scheduled blast SMS event.

Syntax

```
function editAutoConferenceDetails($enterpriseId,
                                   $sessionId,
                                   $UID,
                                   &blastIds,
                                   &$errorText)
```

Description

getBlastSmsList() returns an array of blast identifiers for a scheduled blast SMS events. The system creates these call identifiers when you add recipients to a blast SMS event.

Variables

EnterpriseId	An IntelePeer enterprise ID..
SessionId	A session identifier returned from **requestSession()** function.
UID	The conference requester's user ID. The system assigns the ID when you create the user.

Returns

Upon success, an array of blast identifiers and a 0 (zero). Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed
308	Blast not found

getCallStatus()

Get a call's status.

Syntax

```
function getCallStatus($enterpriseId,
              $sessionId,
              $callIdentifier,
              &$callStatus,
              &$errorText)
```

Description

`getCallStatus()` returns the final status of the completed call specified by the **`callIdentifer`**. If a call has multiple legs, this function returns the final status for each leg. The **`callStatus`** returns a text string detailing the final status.

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`sessionId`	A session identifier returned from **`requestSession()`** function.
`callIdentifier`	A unique call identifier assigned when the call was scheduled.

Returns

On success, returns a string describing the call's final status and a 0 (zero) success. The string contains the following information:

- the user who originated the call
- the call identifier
- the number of minutes used in the call
- the final status of the call --- success or failure.
- the time when the call was scheduled
- the time when the call actually took place
- the time the call completed
- the number that was called

Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

3	Required parameters missing

See Also

`blastVoice()` on page 101.

deleteCall()

Deletes a call scheduled in the future.

Syntax

```
function deleteCall($enterpriseId,
              $sessionId,
              $callIdentifier,
              &$errorText)
```

Description

`deleteCall()` deletes the call specified by the **`callIdentifier`**.

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`sessionId`	A session identifier returned from **`requestSession()`** function.
`callIdentifier`	The unique call identifier assigned when the call was scheduled.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

3	Required parameters missing

deleteBlastVoice()

Deletes a scheduled voice or SMS blast call.

Syntax

```
function deleteBlastVoiceVoice($enterpriseId,
                        $sessionId,
                        $UID,
                        $blastId,
                        &$errorText)
```

Description

`deleteBlastVoiceVoice()` deletes a blast call with the specified **`blastId`**. You must supply a the **`UID`** of the call originator. You can use this method to delete a scheduled voice or SMS blast.

Variables

`enterpriseId`	An IntelePeer enterprise ID.
`sessionId`	A session identifier returned from **`requestSession()`** function.
`blastId`	The blast ID assigned by IntelePeer.
`UID`	The user's ID. The system assigns this value when you call **`createUser()`**.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

304	Internal error: SQL execution failed

getCallIdsByBlastID()

Returns the details of a completed SMS or voice blast call.

Syntax

```
function getCallIdsByBlastID ($enterpriseId,
               $sessionId,
               $blastId,
               $UID,
               &$callIdentifiers,
               &$errorText)
```

Description

getCallIdsByBlastID() retrieves the details of a completed blast call and returns them in an array of **callIdentifiers**. For each termination point, the function returns the following values:

- **callId**
- **mou**
- **number**
- **reason**
- **status**
- **timeScheduled**
- **timeOriginated**
- **timeCompleted**
- **callingNumber**
- **currentState**
- **callComplete**

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
blastId	The blast ID assigned by IntelePeer.

Returns

On success, an array of call identifiers and a 0 (zero) value. The array contains for each Upon failure, this function returns an array containing a non-zero error code and an error message.

Errors

3	Required parameters missing

clearCallIdsByBlastID()

Clears the history call history associated with an SMS or voice blast call.

Syntax

```
function clearCallIdsByBlastID($enterpriseId,
                               $sessionId,
                               $blastId,
                               $UID,
                               &$errorText)
```

Description

clearCallIdsByBlastID() removes the history of a blast call from an user's history. The system maintains history data until you explicitly remove it or the system reaches capacity for history.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
blastId	a unique blast identifier for reference in other calls

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

clearCallIdsByConfID()

Clears the history of a conference call.

Syntax

```
function clearCallIdsByConfID ($enterpriseId,
               $sessionId,
               $confId,
               $UID,
               &$errorText)
```

Description

clearCallIdsByConfID() removes the history of a conference call from an user's history. The system maintains history data until you explicitly remove it or the system reaches capacity for history.

Variables

enterpriseId	An IntelePeer enterprise ID.
sessionId	A session identifier returned from **requestSession()** function.
UID	The user's ID. The system assigns this value when you call **createUser()**.
blastId	A unique conference identifier provided by IntelePeer when you created the conference.

Returns

0 (zero) success. Upon failure, this function returns an array containing a non-zero error code and an error message.

See Also

conferenceOnDemand() on page 93.