CSTAMakeCallConfEventXE "CSTAMakeCallConfEvent"§

The Make Call confirmation event provides the positive response from the server for a previous Make Call service request.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{     ACSHandle_t          acsHandle;EventClass_t     eventClass;       EventType_t
          eventType;
} ACSEventHeader_t;

typedef struct
{
     ACSEventHeader_t    eventHeader;
     union
     {          struct
          {               InvokeID_t       invokeID;              union
               {                    CSTAMakeCallConfEvent_t    makeCall;              } u;
          } cstaConfirmation;
     } event;} CSTAEvent_t;
     typedef struct
     {
          ConnectionID_t         newCall;
     } CSTAMakeCallConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_MAKE_CALL_CONF**, which identifies this message as an

CSTAMakeCallConfEvent.

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

newCall

Specifies the Connection ID for the originating connection of the new call originated by the Make Call request.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaMakePredictiveCall( )XE "cstaMakePredictiveCall( )"§

The **cstaMakePredictiveCall**( ) service originates a call between a group of devices or a logical device on behalf of an originating (calling) device. The service creates a new call and establishes a Connection with the terminating (called) device. The Make Predictive Call service also provides a CSTA Connection Identifier that indicates the Connection of the terminating (called) device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaMakePredictiveCall (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *callingDevice,
      DeviceID_t               *calledDevice,
      AllocationState_t        allocationState;
      PrivateData_t            *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

callingDevice

A pointer to the device identifier of the device which is to originate the new call.

calledDevice

A pointer to the device identifier of the device being call, i.e. the destination device.

allocationState

This parameter specifies under which condition the connection with the destination is to be connected to the calling or originating device. If this parameter is not specified by the application, the Call Delivered state will be the default. This parameter may be one of the following values:

Call Delivered: this value specifies that the switch should attempt to connect the call to the caller (originating device), if the Alerting or Connected state is determined at the called party (destination device).

Call Established: this value specifies that the switch should attempt to connect the call to the caller (originating device), if the Connected state is determined at the called party (destination device).

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never

be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAMakePredictiveCallConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

Comments

This service is often used to initiate a call to a called device (destination) from a group of devices or a logical device without first establishing a connection with a calling device (originator). This service allocates the call to a particular device within that group at some time during the progress of the call.

The **cstaMakePredictiveCall**( ) service first initiates a call to

the called device (destination). Depending on the call's progress, the call may be connected with the calling device (originator) during the progress of the call. The point at which the switch will attempt to connect the call to the originating device is determined by the ***allocation State*** parameter. If the allocation parameter is set to Call Delivered, then the call is allocated upon detection of an Alerting (or Connected) Connection state at the destination. If the allocation parameter is set to Call Established, then the call is allocated upon detection of a Connected Connection state at the recipient. In other words, the call is connected to the originating device if the call state is either alerting or connected at the far end or connected, respectively.

Figure 5-11 illustrates the results of a Make Predictive Call (Calling device = group device D1, Called device = D2.

Figure 5-12

Make Predictive Call Servicetc "Make Predictive Call Service" \f f \l3§
µ §

CSTAMakePredictiveCallConfEventXE "CSTAMakePredictiveCallConfEvent"§

The Make Predictive Call confirmation event provides the positive response from the server for a previous **cstaMakePredictiveCall**( ) request.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{     ACSHandle_t          acsHandle;EventClass_t     eventClass;       EventType_t
          eventType;
} ACSEventHeader_t;

typedef struct
{
      ACSEventHeader_t    eventHeader;
      union
      {           struct
            {                 InvokeID_t       invokeID;                union
                  {                     CSTAMakePredictiveConfEvent_t makePredictiveCall;              } u;          }
      cstaConfirmation;
      } event;} CSTAEvent_t;
      typedef struct
      {
            ConnectionID_t          newCall;
      } CSTAMakePredictiveConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value

MAKE_PREDICTIVE_CALL_CONF, which identifies this message as an **CSTAMakePredictiveConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

newCall

Specifies the Connection ID for the far-end connection of the new call originated by the Make Predictive Call request.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaPickupCall( )XE "cstaPickupCall( )"§

The cstaPickupCall( ) service takes a ringing (alerting) call at a device and redirects the call to a specified device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaPickupCall (
      ACSHandle_t          acsHandle,
      InvokeID_t           invokeID,
      ConnectionID_t       *deflectCall,
      DeviceID_t           *calledDevice,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

deflectCall

This is a pointer to the connection identifier of the call which is to be picked up from another device within the switch.

calledDevice

A pointer to the device identifier of the device which is picking up the original call.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAPickupCallConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL

This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED

This return value indicates that a previously active ACS Stream has been abnormally aborted.

ACSTAERR_REQDENIED

This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

Comments

The **cstaPickupCall**( ) service takes an alerting call at a device within the switch and brings it to a local device destination. This function picks up a call, ***deflectCall,*** at device specified in the ***calledDevice*** parameter.

Figure 5-13

Pickup Call Servicetc "Pickup Call Service" \f f \l3§
µ §

CSTAPickupCallConfEventXE "CSTAPickupCallConfEvent"§

The Pickup Call confirmation event provides the positive response from the server for a previous pickup call request.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {               struct
                {                       InvokeID_t        invokeID;
                        union
                        {
                          CSTAPickupCallConfEvent_t    pickupCall;
                        }u;             } cstaConfirmation;
        } event;} CSTAEvent_t;

typedef struct CSTAPickupCallConfEvent_t {
   Nulltype       null;
} CSTAPickupCallConfEvent_t;
```

Parameters

acsHandle

This is the handle to an active ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_PICKUP_CALL_CONF**,

which identifies this message as an **CSTAPickupCallConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaReconnectCall( )XE "cstaReconnectCall( )"§

The **cstaReconnectCall**( ) service provides the compound action (combination) of the **cstaClearConnection**( ) service followed by the **cstaRetrieveCall**( ) service. The service clears an existing Connection and then retrieves a previously Held Connection at the same device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaReconnectCall (
        ACSHandle_t         acsHandle,
        InvokeID_t          invokeID,
        ConnectionID_t      *activeCall,
        ConnectionID_t      *heldCall,
        PrivateData_t       *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

activeCall

A pointer to the connection identifier of the active call which is to be cleared.

heldCall

A pointer to the connection identifier of the held call which is to

be retrieved.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAReconnectCallConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL

This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED

This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED

This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

Comments

A successful request of this service will causes an existing active call to be dropped. Once the active call has been dropped, the specified held call at the device is retrieved and becomes active. This service is typically used to drop an active call and return to a held call; however, it can also be used to cancel of a consultation call (because of no answer, called device busy, etc.) followed by returning to a held call.

Figure 5-14

Reconnect Call Servicetc "Reconnect Call Service" \f f \l3§
µ §

CSTAReconnectCallConfEventXE "CSTAReconnectCallConfEvent"§

The Reconnect Call confirmation event provides the positive response from the server for a previous Reconnect call request.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {               struct
                {                       InvokeID_t        invokeID;
                        union
                        {
                                CSTAReconnectCallConfEvent_t  reconnectCall;
                        }u;             } cstaConfirmation;
        } event;} CSTAEvent_t;

typedef struct CSTAReconnectCallConfEvent_t {
   Nulltype       null;
} CSTAReconnectCallConfEvent_t;
```

Parameters

acsHandle

This is the handle to an active ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value

CSTA_RECONNECT_CALL_CONF, which identifies this message as an **CSTAReconnectCallConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaRetrieveCall( )XE "cstaRetrieveCall( )"§

The **cstaRetrieveCall**( ) service connects an existing Held Connection. The state of the specified call changes from held to active.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaRetrieveCall (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      ConnectionID_t      *heldCall,
      PrivateData_t       *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

heldCall

A pointer to the connection identifier of the held call which is to be retrieved.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTARetrieveCallConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been

denied by the Client Library Software Driver.

Comments

The indicated held Connection is restored to the Connected state (active). The call state can change depending on the actions of far end endpoints. If the **cstaHoldCall**( ) service reserved the Held Connection and the **cstaRetrieveCall**( ) service is requested for the same call, then the Retrieve Call service uses the reserved Connection.

Figure 5-15

Retrieve Call Servicetc "Retrieve Call Service" \f f \l3§
µ §

CSTARetriveCallConfEventXE "CSTARetriveCallConfEvent"§

The Retrieve Call confirmation event provides the positive response from the server for a previous Retrieve call request.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {               struct
                {                       InvokeID_t        invokeID;
                        union
                        {
                          CSTARetrieveCallConfEvent_t   retrieveCall;
                        }u;             } cstaConfirmation;
        } event;} CSTAEvent_t;

typedef struct CSTARetrieveCallConfEvent_t {
   Nulltype       null;
} CSTARetrieveCallConfEvent_t;
```

Parameters

acsHandle

This is the handle to an active ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value

CSTA_RETRIEVE_CALL_CONF, which identifies this message as an **CSTARetrieveCallConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaTransferCall( )XE "cstaTransferCall( )"§

The **cstaTransferCall**( ) service provides the transfer of a held call with an active call at the same device. The transfer service merges two calls with Connections to a single common device. Also, both of the Connections to the common device become Null and their Connections Identifiers are released.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaTransferCall (
        ACSHandle_t         acsHandle,
        InvokeID_t          invokeID,
        ConnectionID_t      *heldCall,
        ConnectionID_t      *activeCall,
        PrivateData_t       *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

heldCall

A pointer to the connection identifier of the held call which is to be transferred.

activeCall

A pointer to the connection identifier of the active call to which

the held call is to be transferred.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTATransferCallConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL

This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

Comments

Referring to Figure 5-16, the starting conditions for the **cstaTransferCall**( ) service are: the call C1 from D1 to D2 is in held state (***heldCall***). A call C2 from D1 to D3 is in progress or active (***activeCall***). This service transfers the existing (held) call between devices D1 and D2 into a new call with a new call identifier from device D2 to a device D3.

Figure 5-17

Transfer Call Servicetc "Transfer Call Service" \f f \l3§
µ §

The request is used in the situation where the call from D1 to D3 is established (active) or if the call is in any state other than Failed or Null state. The Transfer Call service successfully completes, and D1 is released from the call.

CSTATransferCallConfEventXE "CSTATransferCallConfEvent"§

The Transfer Call confirmation event provides the positive response from the server for a previous transfer call request.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{     ACSHandle_t          acsHandle;EventClass_t     eventClass;       EventType_t
          eventType;
} ACSEventHeader_t;

typedef struct
{
     ACSEventHeader_t    eventHeader;
     union
     {          struct
          {               InvokeID_t       invokeID;              union
               {
                    CSTATransferCallConfEvent_t   transferCall;                  } u;          }
     cstaConfirmation;
     } event;} CSTAEvent_t;


typedef struct Connection_t {    ConnectionID_t  party;    DeviceID_t     staticDevice;  /* NULL for not
     present */} Connection_t;typedef struct ConnectionList {    int          count;    Connection_t
     *connection;} ConnectionList;typedef struct {    ConnectionID_t  newCall;    ConnectionList
     connList;} CSTATransferCallConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_TRANSFER_CALL_CONF**, which identifies this

message as an **CSTATransferCallConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

newCall

Specifies the resulting Connection Identifier for the transferred call.

connList

Specifies the resulting number of known devices in the conference. This field contains a count ***(count)*** of the number of devices in the conference and a pointer ***(*connection)*** to an array of pointers which point to ConnectionID_t structures which define each connection in the call.

Each ConnectionID_t record contains the following:

Party - indicates the Connection ID of the party in the conference.

Device - provides the static reference for the party in the conference. This parameter may have a value that indicates the static identifier is not known.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

Telephony Supplementary ServicesXE "Telephony Supplementary Services"§

This section describes those CSTA telephony services that switches typically provide as "features".

Not all switches support all these functions and events.

Applications can use the Telephony Supplementary Services (defined in this section) to manipulate telephony objects.. As with other TSAPI services already described, these function will generate associated confirmation events from the Telephony Server. Similarly (as described in Chapter 4, ***Sending CSTA Requests and Responses***) applications can use the *invokeID* to match a specific confirmation event with the specific function call, or they may use application-generated *invokeID*s to index into a data structure.

To receive events, an application must have an active ACS Stream and an implement an event handling mechanism. Further, the reception of unsolicited events*XE "Event:Unsolicited"§* requires an active monitor. See the ***Control Services*** and ***Status Reporting Services*** sections for more information on events.

cstaSetMsgWaitingInd( )XE "cstaSetMsgWaitingInd( )"§

The **cstaSetMsgWaitingInd**( ) service provides the application with the ability to turns on and off a message waiting indicator on a telephony device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaSetMsgWaitingInd (
      ACSHandle_t          acsHandle,
      InvokeID_t           invokeID,
      DeviceID_t           *device,
      Boolean_t            messages,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which to set the message waiting indicator. .

msgIndicator

This parameter identifies whether to turn on or off the message waiting indicator at the device specified by ***device*** parameter. A value of TRUE indicates that the message waiting indicator

should be tuned on, FALSE indicates that the indicator should be turn off.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTASetMsgWaitingIndConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL

This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTASetMsgWaitingIndConfEventXE " CSTASetMsgWaitingIndConfEvent"§

The Set Message Waiting Indicator confirmation event provides the positive response from the Telephony Server for a previous Set Message Waiting Indicator service request. When the application receives this event the message waiting indicator has been set as requested by the application.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
        typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
        eventType;
} ACSEventHeader_t;

typedef struct
{
                ACSEventHeader_t    eventHeader;
        union
        {
                struct
                {                       InvokeID_t        invokeID;
                        union
                        {
                                CSTASetMwiConfEvent_t   setMwi;
                        }u;             } cstaConfirmation;
        } event;} CSTAEvent_t;

typedef struct CSTASetMwiConfEvent_t {
   Nulltype       null;
} CSTASetMwiConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which

identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_SET_MWI_CONF**, which identifies this message as an **CSTASetMessageWaitingIndConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaSetDoNotDisturb( )XE "cstaSetDoNotDisturb( )"§

The **cstaSetDoNotDisturb**( ) service activates the switch feature that prevents calls from alerting at a specified device by deflecting the calls from the original destination to other devices.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaSetDoNotDisturb (
      ACSHandle_t          acsHandle,
      InvokeID_t           invokeID,
      DeviceID_t           *device,
      Boolean_t            doNotDisturb,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which the Do Not Disturb feature is to be activated. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

doNotDisturb

This parameter identifies whether to turn on or off the Do Not Disturb feature at the device specified by ***device*** parameter. A value of TRUE indicates that the Do Not Disturb feature should be tuned on, FALSE indicates that the feature should be turn off.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTASetDoNotDisturbConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL

This return value indicates that a bad or unknown

acsHandle was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTASetDoNotDisturbConfEventXE "CSTASetDoNotDisturbConfEvent"§

The Set Do Not Disturb confirmation event provides the positive response from the Telephony Server for a previous Set Do Not Disturb request. When the application receives this event the Do Not Disturb feature has been set as requested by the application.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
                ACSEventHeader_t    eventHeader;
                union
                {
                        struct
                        {                               InvokeID_t        invokeID;
                                union
                                {
                                        CSTASetDndConfEvent_t   setDnd;
                                }u;                     } cstaConfirmation;
                } event;} CSTAEvent_t;

typedef struct CSTASetDndConfEvent_t {
   Nulltype       null;
} CSTASetDndConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_SET_DND_CONF**, which identifies this message as an **CSTASetDoNotDisturbConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaSetForwarding( )XE "cstaSetForwarding( )"§

The **cstaSetForwarding**( ) service activates and deactivates several types of forwarding features on a specified device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaSetForwarding (

      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      ForwardingType_t    forwardingType,
      Boolean_t           forwardingOn,
      DeviceID_t               *forwardingDestination,
      PrivateData_t       *privateData);

typedef enum ForwardingType_t {
   FWD_IMMEDIATE = 0,
   FWD_BUSY = 1,
   FWD_NO_ANS = 2,
   FWD_BUSY_INT = 3,
   FWD_BUSY_EXT = 4,
   FWD_NO_ANS_INT = 5,
   FWD_NO_ANS_EXT = 6
} ForwardingType_t;
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which forwarding is to be set. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

forwardingType

This parameter specifies the type of forwarding to set or clear at the requested device. The possible types include:

Immediate	Forwarding all calls
Busy	Forwarding when busy
No Answer	Forwarding after no answer
Busy Internal	Forwarding when busy for an internal call
Busy External	Forwarding when busy for an external call
No Answer Internal	Forwarding after no answer for an internal call
No Answer External	Forwarding after no answer for an external call.

forwardingOn

This parameter identifies whether to turn on or off the forwarding feature at the device specified by ***device*** parameter.

A value of TRUE indicates that the forwarding feature should be tuned on, FALSE indicates that the feature should be turn off.

forwardingDestination

This is a pointer to the device identifier for the device to which the calls are to be forwarded.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTASetForwardingConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTASetForwardingConfEventXE "CSTASetForwardingConfEvent"§

The Set Forwarding confirmation event provides the positive response from the server for a previous Set Forwarding service request. When this event is received by the application the forwarding feature has been set as requested.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
                struct
                {                       InvokeID_t        invokeID;
                        union
                        {
                                CSTASetFwdConfEvent_t   setFwd;
                        }u;             } cstaConfirmation;
        } event;} CSTAEvent_t;

typedef struct CSTASetFwdConfEvent_t {
   Nulltype       null;
} CSTASetFwdConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_SET_FWD_CONF**, which identifies this message as an **CSTASetForwardingConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaSetAgentState( )XE "cstaSetAgentState( )"§

The **cstaSetAgentState**( ) service changes an ACD agents work mode to one specified by this service.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaSetAgentState (
        ACSHandle_t             acsHandle,
        InvokeID_t              invokeID,
        DeviceID_t              *device,
        AgentMode_t             agentMode,
        AgentID_t         *agentID,
        AgentGroup_t            *agentGroup,
        AgentPassword_t         *agentPassword,
        PrivateData_t           *privateData);

typedef enum AgentMode_t {
   AM_LOG_IN = 0,
   AM_LOG_OUT = 1,
   AM_NOT_READY = 2,
   AM_READY = 3,
   AM_WORK_NOT_READY = 4,
   AM_WORK_READY = 5
} AgentMode_t;

typedef char          AgentID_t[32];
typedef DeviceID_t      AgentGroup_t;
typedef char          AgentPassword_t[32];
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS

Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device associated with the ACD agent for which the work mode is to be changed. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

agentMode

This parameter specifies the work mode state which the agent will be moved to. This could be one of the following:

LOG_INLOG_OUT
NOT_READY
READY
WORK_NOT_READY
WORK_READY

agentID

A pointer to the agent identifier of the ACD Agent whose work mode is to be changed.

agentGroup

A pointer to the agent group identifier for the ACD group or split in which the agent will be logged into or out of. This parameter is only required when the ***agentMode*** parameter is set for the **LOG_IN** and **LOG_OUT** work modes.

agentPassword

A pointer to a password that allows the agent to log into an ACD split or group. This parameter is only required when the ***agentMode*** parameter is set for the **LOG_IN** and **LOG_OUT** work modes.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTASetAgentStateConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been

denied by the Client Library Software Driver.

CSTASetAgentStateConfEventXE "CSTASetAgentStateConfEvent"§

The Set Agent State confirmation event provides the positive response from the server for a previous Set Agent State service request. When this event is received by the application the forwarding feature has been set as requested.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
                struct
                {                       InvokeID_t        invokeID;
                        union
                        {
                                CSTASetAgentStateConfEvent_t   setAgentState;
                        }u;             } cstaConfirmation;
                } event;} CSTAEvent_t;

typedef struct CSTASetAgentStateConfEvent_t {
   Nulltype       null;
} CSTASetAgentStateConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_SET_AGENT_STATE_CONF**, which identifies this message as an **CSTASetAgentStateConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaQueryMsgWaitingInd( )XE "cstaQueryMsgWaitingInd( )"§

The **cstaQueryMessageWaitingInd**( ) service provides the current state of the message waiting indicator of a specified device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaQueryMsgWaitingInd (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which the message waiting indicator is being queried. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

privateData

This is a pointer to the private data extension mechanism.

Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAQueryMsgWaitingIndConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTAQueryMsgWaitingIndConfEventXE "CSTAQueryMsgWaitingIndConfEvent"§

The Query Message Waiting Indicator confirmation event provides the positive response from the server for a previous Query Message Waiting Indicator service request. This event informs the application whether there are any messages waiting, i.e. whether the message waiting indicator is turned on or off.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
                struct
                {                       InvokeID_t        invokeID;                 union
                        {                               CSTAQueryMwiConfEvent_t   queryMwi;               }
u;
                } cstaConfirmation;
        } event;} CSTAEvent_t;

typedef struct CSTAQueryMwiConfEvent_t {
        Boolean         messages;
} CSTAQueryMwiConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which

identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_QUERY_MWI_CONF**, which identifies this message as an **CSTAQueryMwiConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

messages

This parameter specifies whether there are any messages waiting at the requested device. TRUE indicates that there are messages waiting, FALSE indicates that there are none.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaQueryDoNotDisturb( )XE "cstaQueryDoNotDisturb( )"§

The **cstaQueryDoNotDisturb**( ) service provides the current state of the do not disturb feature on a specific device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaQueryDoNotDisturb (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which the Do Not Disturb feature is being queried. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used,

the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAQueryDoNotDisturbConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTAQueryDoNotDisturbConfEventXE "CSTAQueryDoNotDisturbConfEvent"§

The Query Do Not Disturb confirmation event provides the positive response from the server for a previous Query Do Not Disturb service request. This event informs the application whether the feature is turned on or off.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
                        struct
                        {       InvokeID_t        invokeID;  union
                                {               CSTAQueryDndConfEvent_t   queryDnd;    } u;
                        } cstaConfirmation;
                                } event;} CSTAEvent_t;

typedef struct
{
                Boolean_t                doNotDistrub;
        } CSTAQueryDndConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_QUERY_DND_CONF**, which identifies this message as an **CSTAQueryDndConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

doNotDisturb

This parameter specifies whether the Do Not Disturb feature is active at the requested device. TRUE indicates that the feature is turned on. FALSE indicates that the feature is turned off.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaQueryForwarding ( )XE "cstaQueryForwarding ( )"§

The **cstaQueryForwarding**( ) service provides the current state of the forwarding feature(s) on a specific device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaQueryForwarding (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which the forwarding feature is being queried. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used,

the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAQueryForwardingConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTAQueryForwardingConfEventXE "CSTAQueryForwardingConfEvent"§

The Query Forwarding confirmation event provides the positive response from the server for a previous Query Forwarding service request. The event also informs the application of the forwarding type, whether forwarding is on or off, and the forwarding destination for each device requested.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{
      ACSHandle_t          acsHandle;
      EventClass_t     eventClass;
      EventType_t          eventType;
} ACSEventHeader_t;

typedef struct
{
      ACSEventHeader_t    eventHeader;
      union
      {
            struct
            {
                  InvokeID_t       invokeID;
                  union
                  {
                        CSTAQueryFwdConfEvent_t   queryFwd;
                  } u;
            } cstaConfirmation;
      } event;
} CSTAEvent_t;
```

```
typedef enum ForwardingType_t {
        FWD_IMMEDIATE = 0,
        FWD_BUSY = 1,
        FWD_NO_ANS = 2,
        FWD_BUSY_INT = 3,
        FWD_BUSY_EXT = 4,
        FWD_NO_ANS_INT = 5,
        FWD_NO_ANS_EXT = 6
} ForwardingType_t;

typedef struct ForwardingInfo_t {
        ForwardingType_t      forwardingType;
        Boolean                       forwardingOn;
        DeviceID_t                    forwardDN;
} ForwardingInfo_t;

typedef struct ListForwardParameters_t {
        short                 count;
        ForwardingInfo_t      param[7];
} ListForwardParameters_t;

typedef struct CSTAQueryFwdConfEvent_t {
        ListForwardParameters_t    forward;
} CSTAQueryFwdConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_QUERY_FWD_CONF**, which identifies this message as an **CSTAQueryFwdConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

queryfwd

This parameter is a ***ListForwardParameters_t*** structure which contains the following:

count

This parameter indicates how many forwarding list entries are provided. Each entry corresponds to a different device.

param

An array of ForwardingInfo_t structures, each of which is composed of the following elements.

forwardingType

Specifies the type of forwarding set. The types include:

Immediate	Forwarding all calls
Busy	Forwarding when busy
No Answer	Forwarding after no answer
Busy Internal	Forwarding when busy for an internal call
Busy External	Forwarding when busy for an external call
No Answer Internal	Forwarding after no answer for an internal call
No Answer External	Forwarding after no answer for an external call.

forwardingOn

Indicates whether forwarding is active or inactive. TRUE indicates forwarding is active. FALSE indicates forwarding is inactive.

forwardingDN

Specifies the forward-to destination device for the type of forwarding listed.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaQueryAgentState( )XE "cstaQueryAgentState( )"§

The **cstaQueryAgentState**( ) service will provide the application with the current agent state at a device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaQueryAgentState (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which the agent state is being queried. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAQueryAgentStateConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been

denied by the Client Library Software Driver.

CSTAQueryAgentStateConfEventXE "CSTAQueryAgentStateConfEvent"§

The Query Agent State confirmation event provides the positive response from the server for a previous Query Agent State service request. This event will provide the application with the current agent state.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
                        struct
                        {       InvokeID_t        invokeID;  union
                                {         CSTAQueryAgentStateConfEvent_t queryAgentState;       } u;
                        } cstaConfirmation;
                                } event;} CSTAEvent_t;

typedef enum AgentState_t {
   AG_NOT_READY = 0,
   AG_NULL = 1,
   AG_READY = 2,
   AG_WORK_NOT_READY = 3,
   AG_WORK_READY = 4
} AgentState_t;

typedef struct CSTAQueryAgentStateConfEvent_t {
   AgentState_t    agentState;
} CSTAQueryAgentStateConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_QUERY_AGENT_STATE_CONF**, which identifies this message as an **CSTAQueryAgentStateConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

agentState

This parameter specifies the current work mode state of the agent. The possible agent states are:

Null - This indicates that an agent is logged out of the group or device that they serve.

Not Ready - This state indicates that an agent is occupied with some task other than that of serving a call.

Ready - This state indicates that an agent is ready to accept calls.

Work/Not Ready - This state indicates that an agent is occupied with after call work. It also implies that the agent should not receive additional ACD calls.

Work/Ready - This state indicates that an agent is occupied with after call work. It also implies that the agent may receive additional ACD calls.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaQueryLastNumber( )XE "cstaQueryLastNumber( )"§

The **cstaQueryLastNumber**( ) service provides the last number dialed by a specified device.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaQueryLastNumber (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device on which the last number is being queried. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

privateData

This is a pointer to the private data extension mechanism. Setting this parameter is optional. If the parameter is not used,

the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAQueryLastNumberConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTAQueryLastNumberConfEventXE "CSTAQueryLastNumberConfEvent"§

The Query Last Number confirmation event provides the positive response from the server for a previous Query Last Number request. This event provides the last number that was dialed from the requested device.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;       EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
                        struct
                        {       InvokeID_t        invokeID;  union
                                {         CSTAQueryLastNumberConfEvent_t   queryLastNumber;  } u;
                        } cstaConfirmation;
                                } event;} CSTAEvent_t;

typedef struct
{
                DeviceID_t              lastNumber,
        } CSTAQueryLastNumberConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_QUERY_LAST_NUMBER_CONF**, which identifies this message as an **CSTAQueryLastNumberConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

lastNumber

This parameter indicates the last number dialed at the requested device.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.

cstaQueryDeviceInfo( )XE "cstaQueryDeviceInfo( )"§

The **cstaQueryDeviceInfo**( ) service provides general information about a device. The confirmation event for this service will include information on the class and type of device being queried.

Syntax

```
#include <csta.h>
#include <acs.h>

RetCode_t   cstaQueryDeviceInfo (
      ACSHandle_t              acsHandle,
      InvokeID_t               invokeID,
      DeviceID_t               *device,
      PrivateData_t        *privateData);
```

Parameters

acsHandle

This is the value of the unique handle to the opened ACS Stream.

invokeID

A handle provided by the application to be used for matching a specific instance of a function service request with its associated confirmation event. This parameter is only used when the Invoke ID mechanism is set for Application-generated IDs in the **acsOpenStream**( )**.** The parameter is ignored by the ACS Library when the Stream is set for Library-generated invoke IDs.

device

This parameter is a pointer to the device identifier of the device for which information is being requested. This parameter may be different than the originating device depending on the security level defined for the originating device in the Telephony Server.

privateData

This is a pointer to the private data extension mechanism.

Setting this parameter is optional. If the parameter is not used, the pointer should be set to NULL.

Return Values

This function returns the following values depending on whether the application is using library or application-generated invoke identifiers:

Library-generated Identifiers - if the function call completes successfully it will return a positive value, i.e. the invoke identifier. If the call fails a negative error (<0) condition will be returned. For library-generated identifiers the return will never be zero (0).

Application-generated Identifiers - if the function call completes successfully it will return a zero (0) value. If the call fails a negative error (<0) condition will be returned. For application-generated identifiers the return will never be positive (>0).

The application should always check the **CSTAQueryDeviceInfoConfEvent** message to ensure that the service request has been acknowledged and processed by the Telephony Server and the switch.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
This return value indicates that a bad or unknown ***acsHandle*** was provided by the application.

ACSERR_STREAM_FAILED
This return value indicates that a previously active ACS Stream has been abnormally aborted.

CSTAERR_REQDENIED
This return value indicates that a ACS Stream is established but a requested capability has been denied by the Client Library Software Driver.

CSTAQueryDeviceInfoConfEventXE "CSTAQueryDeviceInfoConfEvent"§

The Query Device Info confirmation event provides the positive response from the server for a previous Query Device Info request. This event provides the application with type and class of the requested device.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***ACS Data Types*** and ***CSTA Data Types*** in Section 4 for a complete description of the event structure.

```
typedef struct
{       ACSHandle_t             acsHandle;EventClass_t      eventClass;         EventType_t
                eventType;
} ACSEventHeader_t;

typedef struct
{
        ACSEventHeader_t    eventHeader;
        union
        {
        struct
        {       InvokeID_t          invokeID;  union
                {       CSTAQueryDeviceInfoConfEvent_t queryDeviceInfo;         } u;
        } cstaConfirmation;
                } event;} CSTAEvent_t;

typedef enum DeviceType_t {
   DT_STATION = 0,
   DT_LINE = 1,
   DT_BUTTON = 2,
   DT_ACD = 3,
   DT_TRUNK = 4,
   DT_OPERATOR = 5,
   DT_STATION_GROUP = 16,
   DT_LINE_GROUP = 17,
   DT_BUTTON_GROUP = 18,
   DT_ACD_GROUP = 19,
   DT_TRUNK_GROUP = 20,
   DT_OPERATOR_GROUP = 21,
   DT_OTHER = 255
} DeviceType_t;
```

```
typedef unsigned char   DeviceClass_t;
#define                 DC_VOICE 0x80
#define                 DC_DATA 0x40
#define                 DC_IMAGE 0x20
#define                 DC_OTHER 0x10

typedef struct CSTAQueryDeviceInfoConfEvent_t {
   DeviceID_t      device;
   DeviceType_t    deviceType;
   DeviceClass_t   deviceClass;
} CSTAQueryDeviceInfoConfEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass

This is a tag with the value **CSTACONFIRMATION**, which identifies this message as an CSTA confirmation event.

eventType

This is a tag with the value **CSTA_QUERY_DEVICE_INFO_CONF**, which identifies this message as an **CSTAQueryDeviceInfoConfEvent.**

invokeID

This parameter specifies the function service request instance for the service which was processed at the Telephony Server or at the switch. This identifier is provided to the application when a service request is made.

deviceIdentifier

May provide an alternate short-form static device identifier for the device requested.

deviceType

This parameter indicates the type of device being queried. The possible device types are:

ACD - Automatic Call Distributor (ACD)

ACD group - Automatic Call Distributor (ACD) group

Button - is one instance of a call manipulation point at an individual station.

Button group - is two or more instances of a call manipulation point at an individual station.

Line - is a communications interface to one or more stations.

Line group - is a set of communications interfaces to one or more stations.

Operator - also known as Attendant

Operator group - two or more operator devices used interchangeably or addressed identically.

Other - is any other type for which there is no enumeration defined.

Station - is the traditional telephone device, either simple or featured.

Station group - is two or more stations used interchangeably or addressed identically.

Trunk - a device used to access other switching sub-domains.

Trunk group - typically, two or more trunks providing connections to the same place.

See the ***Functional Call Mode****l* section of this document for more information on device types. Not all switch implementations will support all the device types listed.

deviceClass

This parameter indicates the class of device being queried. The possible device classes are:

Voice - a device that is used to make audio calls. This class includes all normal telephones, as well as computer modems and G3 facsimile machines.

Data - a device that is used to make digital data calls (either circuit switched or packet switched). This class includes computer interfaces and G4 facsimile machines.

Image - a device that is used to make digital data calls involving imaging, or high speed circuit switched data in general. This class includes video telephones and CODECs.

Other - a type of device not covered by data, image, or voice.

See the *Functional Call Model* section of this document for more information on device classes. Not all switch implementations will support all the device classes listed.

privateData

If private data accompanied this event, then the private data would be copied to the location pointed to by the *privateData* pointer in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) function. If the *privateData* pointer is set to NULL in these functions, then no private data will be delivered to the application.