Chapter

7 *SNAPSHOT SERVICESXE "SNAPSHOT SERVICES"§*

An application uses CSTA Snapshot Services to query the current state of a CSTA Call or a Device object. Snapshot services query the switch to provide an application with information about the object. The information is a "*snapshot*" since the state of the Call or Device object changes over time.

The Call Snapshot Services return a list of the Devices and Connections associated with a given Call, and the Connection States for each of those Devices. As Figure 7-1 illustrates, the union of the Connection States for the Call defines the overall Call State. Also refer to the definition of Call State in Chapter 3.

Figure 7-1 shows a Call that has four associated devices. Recall from Chapter 3 that the relationship between a CSTA Call and a CSTA Device is a CSTA Connection (C1, C2, C3, and C4 are Connections). Each Connection has an associated Connection State. The Call Snapshot Services inform an application of each Device that is on a given Call and the associated Connection State for those Devices. The Call State is the union of all the Connection States associated with the Call. The application can use snapshot information to control Connections. For example, if Figure 7-1 shows a four-party conference call, then an application can use the Call Snapshot Services together with the **csta-ClearConnection**( ) service to disconnect any party from the

µ §
conference. To disconnect connection C4, an application uses the Call Snapshot Services to obtain a Connection Identifier (for C4) that it then passes to the **cstaClearConnection**( ) service.

Figure 7-1: Call Snapshot Service

Device Snapshot Services return information about Calls that are associated with a given CSTA Device object. The information includes a list of Calls associated with the given Device and the Connection State of each Call (at that Device). Note the duality here: Call Snapshot Services return information about Connections at all Devices associated with a given Call, while Device Snapshot Services return information about all Connections at a given Device. Applications use the Device Snapshot Services when they need to know what is happening at a specific Device. As Figure 7-2 shows, Device Snapshot Services do not provide information about the other parties on those Calls connected to the given Device.

An application can use Device Snapshot information to manipulate any Connection, (C1, C2, or C3 in Figure 7-2) at the given Device.

µ §
Figure 7-2: Device Snapshot Service
XE "Device Snapshot Service"§

Before an application requests the Call or Device Snapshot Services, it must have previously obtained a Call or Device identifier (that it will use as a parameter to request those services). The identifier specifies a Call or Device in the switching domain. Depending on the implementation, Snapshot Services may not provide information about devices or connections outside of that switching domain (devices not attached to that switch) .

Call Snapshot Services

XE "Call Snapshot Service"§
This section defines the Call Snapshot Services that query the switch for the status of calls within the switching domain. Call Snapshot Services return information about all Devices and Connections associated with a specified CSTA Call object.

cstaSnapshotCallReq( )XE "cstaSnapshotCallReq( )"§

The **cstaSnapshotCallReq**( ) service provides information about a Call object in the switching domain. The service will return the Devices associated with a given Call and the Connection State for each Device. The Call State is the union of the Connection States.

Syntax

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

RetCode_t   cstaSnapshotCallReq (
        ACSHandle_t                 acsHandle,
        InvokeID_t                  invokeID,
        ConnectionID_t        *snapshotObj,
        PrivateData_t          *privateData),
```

Parameters

acsHandle
This is the handle to an active ACS Stream over which the request will be made.

invokeID
This is an application provided handle that the application uses to match a specific instance of a service request with its confirmation event. The application supplies this parameter only when the Invoke ID mechanism is set for Application-generated IDs in **acsOpenStream**(**).** The ACS Library ignores this parameter when the Stream is set for Library-generated invoke IDs.

snapshotObj
This is a pointer to the Connection Identifier identifying the Call object for which Snapshot information is requested.

privateData
This is an optional pointer to CSTA private data.

Return Values

cstaSnapshotCallReq( ) 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, the invoke identifier. If the call fails it will return a negative error (<0). 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 it will return a negative error (<0). For application-generated identifiers the return is never positive (>0).

An application should always check the **CSTASnapshotCallConfEvent** message to insure that the Telephony Server and switch have acknowledged and processed the **cstaSnapshotCallReq**( ) request.

The following are possible negative error conditions for this function:

ACSERR_BADHDL
The application provided a bad or unknown ***acsHandle.***

ACSERR_STREAM_FAILED
A previously active ACS Stream has been abnormally aborted.

Comments

A call to **cstaSnapshotCallReq**( ) results in a confirmation event, **CSTASnapshotCallConfEvent,** that returns the information about the call. **cstaSnapshotCallReq**( ) provides information about calls that make further monitoring meaningful. For example, when an application requests **cstaMonitorStart**(**),** there may already be active calls at the Device being monitored. The application can use Call Snapshot information to obtain information about those existing calls process additional events about them in a reasonable way.

cstaSnapshotCallReq( ) is passive and does not affect the state of any object in the switching domain.

CSTASnapshotCallConfEventXE "CSTASnapshotCallConfEvent"§

The Call Snapshot confirmation event returns call related information in response to the **cstaSnapshotCallReq**( ) service. The call information includes the static Device Identifiers, the Connection Identifiers, and Connection States for every endpoint in the specified call.

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
			{
				CSTASnapshotCallConfEvent_t  snapshotCall;
			} u;		} cstaConfirmation
	} event;} CSTAEvent_t;

typedef struct CSTASnapshotCallConfEvent_t {	CSTASnapshotCallData_t
snapshotData;} CSTASnapshotCallConfEvent_t;
typedef struct CSTASnapshotCallData_t {	int			count;	struct
	CSTASnapshotCallResponseInfo_t *info;} CSTASnapshotCallData_t;
typedef struct CSTASnapshotCallResponseInfo_t {	SubjectDeviceID_t
	deviceOnCall;	ConnectionID_t			callIdentifier;
	LocalConnectionState_t		localConnectionState;}
CSTASnapshotCallResponseInfoEvent_t;
```

Parameters

acsHandle

This is the handle for the ACS Stream over which the confirmation arrived. This is the same as the ACS Stream over which the request was made.

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_SNAPSHOT_CALL_CONF**, which identifies this message as an **CSTASnapshotCallConfEvent.**

invokeID

This parameter specifies the service request instance for the **cstaSnapshotCallReq**( ) The application uses this parameter to correlate responses with requests.

snapshotData
Contains all the snapshot information for the Call for which the query was made.

> ***count***
> A count of the number of *CSTASnapshotCallResponseInfo_t* structures. Each structure contains information about one device on the call.
>
> ***info***
> A pointer to an array of *CSTASnapshotCallResponseInfo_t* structures, each of which contains the following fields:
>
> > ***deviceOnCall***
> > A pointer to the Device Identifier of a device that is a party on the call for which the query was made.
> >
> > ***callIdentifier***
> > The Connection Identifier for the Connection between the ***deviceOnCall*** and the call for which the query was made.
> >
> > ***localConnectionState***
> > The Connection State for the local Connection in the ***callIdentifier*** parameter.

privateData
If private data accompanies this event, then the private data would be stored in the location that the application specified as the *privateData* parameter in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) request. If the *privateData* pointer is set to NULL in these requests, then **CSTASnapshotCallConfEvent** does not deliver private data to the application.

Comments

The **CSTASnapshotCallConfEvent** returns a linked list since the number of devices on a call can be greater than one. Each member of the list identifies a Device on the call, the Connection between the Device and the Call, and the Connection State (see Figure 7-1). An application should be aware that the number of members on the list is not fixed. The pointer, **next*, will be *NULL* for the last member (device) on the list.

Device Snapshot ServiceXE "Device Snapshot Service"§

This section defines the Device Snapshot Services that query the switch for the status of Devices within the switching domain. Device Snapshot Services return information about Calls (Connections) associated with a specified Device.

cstaSnapshotDeviceReq( )XE "cstaSnapshotDeviceReq( )"§

The **cstaSnapshotDeviceReq**( ) service returns information about a Device object in the switching domain. The service returns a list of calls associated with the given Device and the Connection State of each of those calls at that Device.

Syntax

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

RetCode_t   cstaSnapshotDeviceReq (
            ACSHandle_t             acsHandle,
            InvokeID_t              invokeID,
            DeviceID_t              *snapshotObj,
            PrivateData_t           *privateData);
```

Parameters

acsHandle
This is the handle to an active ACS Stream over which the request will be made.

invokeID
This is an application provided handle that the application uses to match a specific instance of a service request with its confirmation event. The application supplies this parameter only when the Invoke ID mechanism is set for Application-generated IDs in **acsOpenStream**(**).** The ACS Library ignores this parameter when the Stream is set for Library-generated invoke IDs.

snapshotObj
This parameter is a pointer to the Device Identifier for the Device object for which Snapshot information is being requested.

privateData

This is an optional pointer to CSTA private data.

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, the invoke identifier. If the call fails it will return a negative error (<0). 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 it will return a negative error (<0). For application-generated identifiers the return is never positive (>0).

The application should always check the **CSTASnapshotDeviceConfEvent** message to insure that the Telephony Server and the switch have acknowledged and processed the **cstaSnapshotDeviceReq**( ) request.

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.

Comments

A call to **cstaSnapshotDeviceReq**( ) results in a confirmation event, **CSTASnapshotDeviceConfEvent**, which returns information about the Device. **cstaSnapshotDeviceReq**( ) provides information about Devices that permit an application to synchronize state with the switching domain. For example, an application can call **cstaSnapshotDeviceReq**( ) to find out about the Calls present at a Device, then call **cstaMonitorStart**( ) to monitor the Device. The information from the Device query permits the application to process the monitoring events in a proper context.

The **cstaSnapshotDeviceReq**( ) is passive and does not affect the state of any object within the switching domain.

CSTASnapshotDeviceConfEventXE "CSTASnapshotDeviceConfEvent"§

The Device Snapshot confirmation event returns Device related information in response to the **cstaSnapshotDeviceReq**( ) service. The Device information includes a Connection Identifier for each Call at the Device and the Connection State for each Call at the 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
                  {
                        CSTASnapshotDeviceConfEvent_t      snapshotDevice;
                  } u;          } cstaConfirmation;
      } event;} CSTAEvent_t;
typedef struct CSTASnapshotDeviceConfEvent_t {       CSTASnapshotDeviceData_t
snapshotData;}                 CSTASnapshotDeviceConfEvent_t;typedef                 struct
CSTASnapshotDeviceData_t {    int            count;       struct
      CSTASnapshotDeviceResponseInfo_t *info;} CSTASnapshotDeviceData_t;
typedef struct CSTASnapshotDeviceResponseInfo_t {   ConnectionID_t
      callIdentifier;    CSTACallState_t       callState;}
CSTASnapshotDeviceResponseInfo_t;typedef struct CSTACallState_t { int
                              count;       LocalConnectionState_t             *state;}
CSTACallState_t;typedef enum CSTASimpleCallState_t {     CALL_NULL        =        0,
      CALL_PENDING = 1,       CALL_ORIGINATED = 3, CALL_DELIVERED  =  35,
      CALL_DELIVERED_HELD = 36,      CALL_RECEIVED               =              50,
      CALL_ESTABLISHED = 51,      CALL_ESTABLISHED_HELD          =          52,
      CALL_RECEIVED_ON_HOLD = 66, CALL_ESTABLISHED_ON_HOLD  =  67,
      CALL_QUEUED = 83,       CALL_QUEUED_HELD = 84,   CALL_FAILED = 99,
      CALL_FAILED_HELD = 100} CSTASimpleCallState_t;
      /* Used to take a CSTACallState_t which contains only two
 * LocalConnectionState_t and match them to the set of
 */    #define SIMPLE_CALL_STATE(ccs) (ccs.stat[0]+(ccs.state[1] << 4))
typedef struct CSTACallState_t {
            int                                          count;
            LocalConnectionState_t            *state;
} CSTACallState_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream over which the confirmation arrived. This is the same as the ACS Stream over which the request was made.

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_SNAPSHOT_DEVICE_CONF**, which identifies

this message as an **CSTASnapshotDeviceConfEvent.**

invokeID
This parameter specifies the service request instance for the **cstaSnapshotDeviceReq**( ) The application uses this parameter to correlate responses with requests.

snapshotData
Contains all the snapshot information for the Device for which the query was made.

count
A count of the number of *CSTASnapshotDeviceResponseInfo_t* structures. Each contains information about one Device on the Call.

info
A pointer to an array of *CSTASnapshotDeviceResponseInfo_t* structures, each of which contains the following fields:

callIdentifier
A pointer to a Connection Identifier for each call at the device. For some implementations, this parameter points to the device's dynamic device identifier for the call object.

callState
The CSTA Call State. The Call State is returned as a list of local Call States. If there are only two Call States, then ***count*** is two. The application can use the macro ***SIMPLE_CALL_STATE***( ) to determine if a local Call State is one of the CSTA Simple Call States (defined in Chapter 3 and

enumerated in the ***CSTASimpleCallState_t*** structure).

privateData
If private data accompanies this event, then the private data would be stored in the location that the application specified as the *privateData* parameter in the **acsGetEventBlock**( ) or **acsGetEventPoll**( ) request. If the *privateData* pointer is set to NULL in these requests, then **CSTASnapshotDeviceConfEvent** does not deliver private data to the application.

Comments

The **CSTASnapshotDeviceConfEvent** returns a linked list since the number of calls on a device can be greater than one. Each member of the list identifies a call at the device, and the local call state of the Connection for that call at the device (see Figure 7-2). An application should be aware that the number of members on the list is not fixed. The pointer, **next*, will be *NULL* for the last member (call) on the list.