CSTATransferredEventXE "CSTATransferredEvent"§

This event report indicates that an existing call was transferred to another device and that the device which transferred the call is no longer part of the call, i.e. the transferring device has dropped from the call.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***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
                {                       CSTAMonitorCrossRefID_t     monitorCrossRefID;
                            union
                            {
                                        CSTATransferEvent_t  transferred;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        ConnectionID_t                  primaryOldCall;
        ConnectionID_t                  secondaryOldCall;
        SubjectDeviceID_t           transferringDevice;
        SubjectDeviceID_t           transferredDevice;
        ConnectionList_t            transferredConnections;
        LocalConnectionState_t      localConnectionInfo;
        CSTAEventCause_t            cause;
} CSTATransferredEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass

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

eventType

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

monitorCrossRefID

This parameter contains the handle to the CSTA association for which this event is associated. This handle is typically chosen by the switch and should be used by the application as a reference to a specific established association.

primaryOldCall

This parameter identifies the primary known call that was transferred.

secondaryOldCall

This parameter identifies the secondary call that was transferred. This would identify the consultative call used to make the transfer, after the primary call was placed on hold.

transferringDevice

This indicates which device transferred the call. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

transferredDevice

This indicates to which device the call was transferred. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

transferredConnections

This is a list of connections (parties) on the call which resulted from the transfer. The call ID may be different from either the primary or secondary old call (or both)..

localConnectionInfo

This parameter defines the local connection state of the call after the calls have been transferred from the device which performed the transfer. This could be null, initiated, alerting, connected, held, queued, or failed.

cause

This parameter contains the cause value which indicates the reason or explanation for the occurrence of this event. The possible events are defined by **CSTAEventCause_t**.

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.

Comments

This event provides the application with all the information it needs regarding a call which was transferred from one device to another.

Before After

Figure 1 - Transferred Event Report

Feature Event Reports (Unsolicited)XE "Feature Event Reports (Unsolicited)"§

This section covers event reports which pertain to the use of features supported through this API. The feature event reports indicate a change in the state of a specific feature which is operating on a call or a device on the switch. Each event defines the current state of the feature regardless of what the state of the feature was before a feature event is received.

CSTACallInfoEventXE "CSTACallInfoEvent"§

This event report is provided when a user account code feature has collected data for a party on the call. The event includes the account code and authorization information which was collected by the switch feature.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***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
                {                       CSTAMonitorCrossRefID_t     monitorCrossRefID;
                            union
                            {
                                        CSTACallInfoEvent_t  callInformation;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
        {
                ConnectionID_t              connection;
                SubjectDeviceID_t   device;
                AccountInfo_t               accountInfo;
                AuthCode_t                  authorizationCode;
        } CSTACallInforEvent_t;

                typedef char        AccountInfo_t[32];

        typedef char        AuthCode_t[32];
```

Parameters

acsHandle

This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType
This is a tag with the value **CSTA_CALL_INFORMATION**, which identifies this message as an **CSTACallInfoEvent.**

monitorCrossRefID
This parameter contains the handle to the CSTA association for which this event is associated. This handle is typically chosen by the switch and should be used by the application as a reference to a specific established association.

connection
This parameter identifies the party that has entered the account code.

device
Indicates from which device was the account code information entered. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

accountInfo
Specifies the account code which was entered at the device.

authorizationCode
Specifies the authorization code which was enetered at the 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.

Comments

This event informs the application when an account code feature has been activated and what information was collected by the switch as a result of the feature being activated.

CSTADoNotDisturbEventXE "CSTADoNotDisturbEvent"§

This event report indicates a change in the status of the Do Not Disturb feature for a specific device. The Do Not Disturb event will result in all calls to a device to be automatically forwarded to the device coverage path.

Syntax

The following structure shows only the relevant portions of the unions for this message. See s***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
                {                       CSTAMonitorCrossRefID_t     monitorCrossRefID;
                            CSTAEventCategory_t             eventCategory;
                            union
                            {
                              CSTADoNotDisturbEvent_t  doNotDisturb,
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t   device;
        Boolean                     doNotDisturbOn;
} CSTADoNotDisturbEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType
This is a tag with the value **CSTA_DO_NOT_DISTURB**, which identifies this message as an **CSTADoNotDisturbEvent.**

monitorCrossRefID
This parameter contains the handle to the CSTA association for which this event is associated. This handle is typically chosen by the switch and should be used by the application as a reference to a specific established association.

device
Specifies the device for which the DO Not Disturb feature has been activated/deactivated. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

doNotDisturbON
Specifies whether the DO Not Disturb feature is on (1) or off (0).

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.

CSTAForwardingEventXE "CSTAForwardingEvent"§

This event report will indicate a change in the state of the Forwarding feature for a specific device. The event will also indicate the type of forwarding being invoked when the feature is activated.

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
                {                   CSTAMonitorCrossRefID_t     monitorCrossRefID;
                            union
                            {
                                    CSTAForwardingEvent_t  forwarding;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t   device;
        ForwardingInfo_t    forwardingInformation;
} CSTAForwardingEvent_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;
/* NULL for not present */} ForwardingInfo_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType
This is a tag with the value **CSTA_FORWARDING** which identifies this message as an **CSTAForwardingEvent.**

monitorCrossRefID
This parameter contains the handle to the CSTA association for which this event is associated. This handle is typically chosen by the switch and should be used by the application as a reference to a specific established association.

device
Specifies the device for which the Forwarding feature has been activated/deactivated. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

forwardingType
Specifies the type of forwarding being invoked for the specific device. This may include one of the following:

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 afterno answer for an internal call
No Answer External	Forwarding after no answer for an external call.

forwardingON

Specifies whether the Forward feature is on (1) or off (0).

forwardDN

Specifies the destination device to which the calls are being forwarded. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

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.

Comments

The application should be aware that the ***forwardingInfo*** paramater can indicate any of the defined values depending on the switch implementation of the forwarding feature.

CSTAMessageWaitingEventXE "CSTAMessageWaitingEvent"§

This event report is used to indicate whether the Message Waiting feature has been activated/deactivated.

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
                {                       CSTAMonitorCrossRefID_t    monitorCrossRefID;
                            union
                            {
                              CSTAMessageWaitingEvent_t  messageWaiting;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        CalledDeviceID_t    deviceForMessage;
        SubjectDeviceID_t   invokingDevice;
        Boolean                     messageWaitingOn;
} CSTAMessageWaitingEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType

This is a tag with the value **CSTA_MESSAGE_WAITING** which identifies this message as an **CSTAMessageWaitingEvent.**

monitorCrossRefID

This parameter contains the handle to the CSTA association for which this event is associated. This handle is typically chosen by the switch and should be used by the application as a reference to a specific established association.

deviceForMessage

Indicates the device where the message is waiting (i.e. address of device where the message waiting feature was activated). If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

invokingDevice

Specifies which device invoked the message waiting feature (i.e. address of the device who activated the message waiting feature). If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

messageWaitingOn

Specifies whether the Message Waiting feature is on (1) or off (0).

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.

Comments

This event can occur for both a device or a call association.

Agent Status Event Reports (Unsolicited)

XE "Agent Status Event Reports (Unsolicited)"§
This section covers event reports which pertain to the use of ACD agent features supported through the API. The agent feature event reports indicate a change in the state of a specific agent. Each event defines the current state of the feature regardless of what the state of the feature was before a feature event is received. These event will typically be used by applications in the call center or message center environment. Standard desktop telephony applications will typically not utilize this feature of the API.

CSTALoggedOnEventXE "CSTALoggedOnEvent"§

This event report informs the application that an agent has logged into a device (usually an ACD Split).

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
                {                       CSTAMonitorCrossRefID_t    monitorCrossRefID;
                            union
                            {
                                    CSTALoggedOnEvent_t  loggedOn,
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t   agentDevice;
        AgentID_t           agentID;
        AgentGroup_t                agentGroup;
        AgentPassword_t     password;
} CSTALoggedOnEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType

This is a tag with the value **CSTA_LOGGED_ON** which identifies this message as an **CSTALoggedOnEvent.**

monitorCrossRefID

This parameter contains the handle to the CSTA association for which this event is associated. This handle is typically chosen by the switch and should be used by the application as a reference to a specific established association.

agentDevice

Specifies the device from which the agent is logged on to the system. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

agentID

This paramater specifies the agent identifier of the agent who logged into the system.

agentGroup

Specifies the group or ACD Split to which the agent is logging into.

password

This paramater specifies the agent's password used to log into the system.

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.

Comments

In most cases, when an agent logs into a device it usuallty means that the agent is ready to start receiving calls at the device. This may not be true for some implementations.

CSTALoggedOffEventXE "CSTALoggedOffEvent"§

This event report indicates that an agent has logged out of the device/ACD Split for which the agent had previously logged in and was providing service.

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
                {                       CSTAMonitorCrossRefID_t     monitorCrossRefID;
                            union
                            {
                                        CSTALoggedOffEvent_t  loggedOff;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t           agentDevice;
        AgentID_t                   agentID;
        AgentGroup_t                        agentGroup;
} CSTALoggedOffEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which

identifies this message as an CSTA unsolicited event.

eventType
This is a tag with the value **CSTA_LOGGED_OFF** which identifies this message as an **CSTALoggedOffEvent.**

agentDevice
Specifies the device from which the agent is logged off the system. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

agentID
This paramater specifies the agent identifier of the agent who logged off the system.

agentGroup
Specifies the group or ACD Split from which the agent is logging out.

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.

CSTANotReadyEventXE "CSTANotReadyEvent"§

This event report indicates that an agent is busy with tasks other than servicing a call at the device. In most cases this will imply that the agent is not ready to recieve a call or that the agent is taking a break.

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
                {                       CSTAMonitorCrossRefID_t    monitorCrossRefID;
                            union
                            {
                                        CSTANotReadyEvent_t  notReady;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t               agentDevice;
        AgentID_t                       agentID;
} CSTANotReadyEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which

identifies this message as an CSTA unsolicited event.

eventType

This is a tag with the value **CSTA_NOT_READY** which identifies this message as an **CSTANotReadyEvent.**

agentDevice

Specifies the device from which the agent is logged on to the system. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

agentID

This paramater specifies the identifier of the agent who in not ready to receive 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.

CSTAReadyEventXE "CSTAReadyEvent"§

This event report indicates that an agent is ready to receive calls at the device. This event can occur even if the agent is busy on an active call at the device.

Syntax

The following structure shows only the relevant portions of the unions for this message. See ***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
                {                       CSTAMonitorCrossRefID_t    monitorCrossRefID;
                            union
                            {
                                    CSTAReadyEvent_t  ready;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t           agentDevice;
        AgentID_t                   agentID;
} CSTAReadyEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType

This is a tag with the value **CSTA_READY** which identifies this message as an **CSTAReadyEvent.**

agentDevice

Specifies the device which is ready to receive calls from the ACD. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

agentID

This paramater specifies the identifier of the agent who in ready to receive 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.

CSTAWorkNotReadyEventXE "CSTAWorkNotReadyEvent"§

This event report indicates that the agent is in after call work mode completing the tasks involved in servicing a call after the connection has been disconnected. This will implies that the agents is no longer on the call but is completing the servicing of the last call and the agent ***should not*** receive any additional calls.

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
                {                       CSTAMonitorCrossRefID_t    monitorCrossRefID;
                            union
                            {
                               CSTAWorkNotReadyEvent_t  workNotReady;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t           agentDevice;
        AgentID_t                   agentID;
} CSTAWorkNotReadyEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType
This is a tag with the value **CSTA_WORK_NOT_READY** which identifies this message as an **CSTAWorkNotReadyEvent.**

agentDevice
Specifies the device which has invoked the Work Not Ready mode. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

agentID
This paramater specifies the identifier of the agent who is in the Work Not Ready mode.

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.

Comments

In the case of this event the agent is still working on completing the after call work for the last call. The difference between this event and the **CSTAWorkReadyEvent** is that the agent has indicated that he/she is not ready to receive additional calls.

CSTAWorkReadyEventXE "CSTAWorkReadyEvent"§

This event report indicates that the agent is in "after call work mode" completing the tasks involved in servicing a call after the connection has been disconnected. This implies that the agents is no longer on the call but is completing the servicing of the last call and the agent ***may*** receive any additional calls.

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
                {                       CSTAMonitorCrossRefID_t    monitorCrossRefID;
                            union
                            {
                                        CSTAWorkReadyEvent_t  workReady;
                            } u;
                } cstaUnsolicited;
        } event;} CSTAEvent_t;
typedef struct
{
        SubjectDeviceID_t           agentDevice;
        AgentID_t                   agentID;
} CSTAWorkReadyEvent_t;
```

Parameters

acsHandle
This is the handle for the ACS Stream.

eventClass
This is a tag with the value **CSTAUNSOLICITED**, which identifies this message as an CSTA unsolicited event.

eventType
This is a tag with the value **CSTA_WORK_READY** which identifies this message as an **CSTAWorkReadyEvent.**

agentDevice
Specifies the device which has invoked the Work Ready mode. If the device is not specified, then the parameter will indicate that the device was not known or that it was not required.

agentID
This paramater specifies the identifier of the agent who is in the Work Ready mode.

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.

Comments

In the case of this event the agent is still working on completing the after call work for the last call. The difference between this event and the CSTAWorkNotReadyEvent is that the agent has indicated that he/she is ready to receive additional calls.

Event Report Data Types (Unsolicited)XE "Event Report Data Types (Unsolicited)"§

This section defines the data structures associated with the CSTA Event Reports defined in the "STATUS REPORTING SERVICES" section of this document.

CSTAMonitorFilter_tXE "CSTAMonitorFilter_t"§

This structure is used to identify the event type filters requested or available on a monitored CSTA association.

```
typedef unsigned short CSTACallFilter_t;#define CF_CALL_CLEARED
0x8000#define CF_CONFERENCED 0x4000#define
CF_CONNECTION_CLEARED 0x2000#define CF_DELIVERED
0x1000#define CF_DIVERTED 0x0800#define
CF_ESTABLISHED 0x0400#define CF_FAILED 0x0200#define
CF_HELD 0x0100#define CF_NETWORK_REACHED 0x0080#define
CF_ORIGINATED 0x0040#define CF_QUEUED 0x0020#define
CF_RETRIEVED 0x0010#define CF_SERVICE_INITIATED
0x0008#define CF_TRANSFERRED 0x0004typedef unsigned char
CSTAFeatureFilter_t;#define FF_CALL_INFORMATION 0x80#define
FF_DO_NOT_DISTURB 0x40#define FF_FORWARDING 0x20#define
FF_MESSAGE_WAITING 0x10typedef unsigned char CSTAAgentFilter_t;#define
AF_LOGGED_ON 0x80#define AF_LOGGED_OFF 0x40#define
AF_NOT_READY 0x20#define AF_READY 0x10#define
AF_WORK_NOT_READY 0x08#define AF_WORK_READY 0x04typedef
unsigned char CSTAMaintenanceFilter_t;#define
MF_BACK_IN_SERVICE 0x80#define MF_OUT_OF_SERVICE
0x40typedef struct CSTAMonitorFilter_t { CSTACallFilter_t call;
CSTAFeatureFilter_t feature; CSTAAgentFilter_t agent; CSTAMaintenanceFilter_t
maintenance; Boolean private;} CSTAMonitorFilter_t;
```

CALL_FILTERS

These values indicate that a call event filter should be used for processing events. The provided filter may be different than the one requested.

FEATURE_FILTERS

These values indicate that a feature event filter should be used for processing events. The provided filter may be different than the one requested.

AGENT_FILTERS

These values indicate that a agent event filter should be used for processing events. The provided filter may be different than the one requested.

MAINTENANCE_FILTERS
These values indicate that a maintenance event filter should be used for processing events. The provided filter may be different than the one requested.

PRIVATE_FILTER
This value indicates that a private filter should be used for processing events. The provided filter may be different than the one requested.

CSTAEventCause_tXE "CSTAEventCause_t"§

This structure contains an enumerated list of all the possible event causes which can occur with different events. The definitions of these event cause codes are also provided.

```
typedef enum CSTAEventCause_t {
   EC_NONE = -1,
   EC_ACTIVE_MONITOR = 1,
   EC_ALTERNATE = 2,
   EC_BUSY = 3,
   EC_CALL_BACK = 4,
   EC_CALL_CANCELLED = 5,
   EC_CALL_FORWARD_ALWAYS = 6,
   EC_CALL_FORWARD_BUSY = 7,
   EC_CALL_FORWARD_NO_ANSWER = 8,
   EC_CALL_FORWARD = 9,
   EC_CALL_NOT_ANSWERED = 10,
   EC_CALL_PICKUP = 11,
   EC_CAMP_ON = 12,
   EC_DEST_NOT_OBTAINABLE = 13,
   EC_DO_NOT_DISTURB = 14,
   EC_INCOMPATIBLE_DESTINATION = 15,
   EC_INVALID_ACCOUNT_CODE = 16,
   EC_KEY_CONFERENCE = 17,
   EC_LOCKOUT = 18,
   EC_MAINTENANCE = 19,
   EC_NETWORK_CONGESTION = 20,
   EC_NETWORK_NOT_OBTAINABLE = 21,
   EC_NEW_CALL = 22,
   EC_NO_AVAILABLE_AGENTS = 23,
   EC_OVERRIDE = 24,
   EC_PARK = 25,
   EC_OVERFLOW = 26,
   EC_RECALL = 27,
   EC_REDIRECTED = 28,
   EC_REORDER_TONE = 29,
   EC_RESOURCES_NOT_AVAILABLE = 30,
   EC_SILENT_MONITOR = 31,
   EC_TRANSFER = 32,
   EC_TRUNKS_BUSY = 33,
   EC_VOICE_UNIT_INITIATOR = 34
} CSTAEventCause_t;
```

Certain cause codes will appear in events only if they make sense. The table following the cause code definitionsXE "cause code definitions"§ illustrates which cause codes are possible for the each of the call events.

Cause Code	Indicates that (definition):
Active Monitor	an Active Monitor Feature has occurred. This feature typically allows intrusion by a supervisor into an agent call with the ability to speak and listen. The resultant call can be considered as a conference so this cause code may be supplied with the Conferenced Event Report.
Alternate	the call is in the process of being exchanged. This feature is typically found on single-line telephones, where the human interface puts one call on hold and retrieves a held call or answers a waiting call in an atomic action.
Busy	the call encountered a busy tone or device
Call Back	Call Back is a feature invoked (by a user or via CSTA) in an attempt to complete a call that has encountered a busy or no answer condition. As a result of invoking the feature, the failed call is cleared and the call can be considered as queued. The switch may subsequently automatically retry the call (normally when the called party next becomes free). Consequently, this cause code may appear in Event

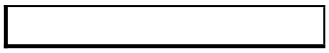
	Reports related to the feature invocation (Call Cleared, Connection Cleared and Queued) or related to the subsequent, retried call (Service Initiated, Originated, Delivered, and Established).
Call Cancelled	the user has terminated a call without going on-hook.
Call Forward	the call has been redirected via a Call Forwarding feature set for general, unknown, or multiple conditions.
Call Fd. - Immediate	the call has been redirected via a Call Forwarding feature set for all conditions.
Call Fd. - Busy	the call has been redirected via a Call Forwarding feature set for a busy endpoint.
Call Fd. - No Answer	the call has been redirected via a Call Forwarding feature set for an endpoint that does not answer.
Call Not Answered	the call was not answered because a timer has elapsed.
Call Pickup	the call has been redirected via a Call Pickup feature.
Camp On	a Camp On feature has been invoked or has matured.
Dest. Not Obtainable	the call could not obtain the destination.

Do Not Disturb	the call encountered a Do Not Disturb condition.
Incompatible Destination	the call encountered an incompatible destination.
Invalid Account Code	the call has an invalid account code.
Key Operation[1]	indicates that the Event Report occurred at a bridged or twin device.
Lockout	the call encountered inter-digit timeout while dialing.
Maintenance	the call encountered a facility or endpoint in a maintenance condition.
Net Congestion	the call encountered a congested network. In some circumstances this cause code indicates that the user is listening to a "No Circuit" Special Information Tone (SIT) from a network that is accompanied by a statement similar to "All circuits are busy..."
Net Not Obtainable	the call could not reach a destination network.
New Call	the call has not yet been redirected.

1 *Telephone numbers associated primarily with one device often appear also on a second device. One example is a secretary who's phone has mirrored or bridged lines of a boss's phone.*

No Available Agents	the call could not access any agent.
Overflow	the call overflowed a queue, group, or target.
Override	the call resulted because of an Override feature.
Park	indicate that the Event Report is associated with an action to place a call to or retrieve a call from a parked position. Placing a call in a park position releases the call from the parking device, but retains the call in the Switching Function so that it can be connected to another (or the same) device by invoking the un-parking feature there.
Recall	the call is alerting a device due to a time-out built into a feature that failed to complete or that anticipated further action from the user.
Redirected	the call has been redirected
Reorder Tone	the call encountered reorder - a tone provided by a network to indicate that the request (call, feature, or supplementary service) was not recognizable. This condition usually results when a user dials a number that is not valid or attempts to obtain a service that is not enabled for that user

	or device. In some circumstances this cause code indicates that the user is listening to a "Reorder" Special Information Tone (SIT) from a network that is accompanied by a statement similar to "The call did not go through as dialed..."
Resources not Available	resources were not available
Silent Monitor	the event was caused by the invocation of a feature that allows a third party, such as an ACD agent supervisor, to join the call. The joining party can hear the entire conversation, but cannot be heard by either original party. The feature, sometimes called *silent intrusion*, may provide a tone to one or both parties to indicate that they are being monitored. This feature is not the same as a CSTA Monitor request. This cause shall not indicate that a CSTA Monitor has been initiated.
Transfer	a Transfer is in progress or has occurred
Trunks Busy	the call encountered Trunks Busy
Voice Unit Initiator	indicates that the event was the result of action by automated equipment (voice mail device, voice response unit, announcement) rather than the result of action by a human user.



The following table illustrates which cause codes values make sense for a specific call status event. The "y" indicates that the cause code is likely to appear in the specific event.

Table 6-2 CSTA Event Report - Cause Relationships

Cause	Call Clr.	Conf	Con. Clr.	Dlv.	Div.	Est.	Fail	Held	Net. Rch.	Orig.	Q-ed	Retr.	Svc. Init.	Tran
Active Monitor		y												
Alternate						y	y	y				y		
Busy							y				y			
Call Back	y		y	y						y	y		y	
Call Cancelled	y		y				y						y	
Call Forward				y	y		y	y	y		y			
Call Fd. - Immediate				y	y		y		y		y			
Call Fd. - Busy				y	y		y		y		y			
Call Fd. - No Answer				y	y		y	y	y		y			
Call Not Answered	y		y		y		y							
Call Pickup					y	y								
Camp On				y			y				y			
Dest. not Obtainable			y				y				y			
Do Not Disturb			y		y		y				y			
Incpt. Destination	y		y		y		y							
Invalid Account Code	y						y							
Key Operation	y	y	y	y	y	y	y	y	y	y	y	y	y	y
Lockout							y							
Maintenance	y						y							
Net Congestion							y				y			
Net Not Obtainable							y				y			
New Call		y		y		y				y				y
No Available Agents				y	y		y				y			
Overflow	y		y	y	y		y		y		y			

Override	y	y	y	y		y	y			y			y	
Park			y								y			
Recall		y		y	y	y	y	y			y	y		y
Redirected				y	y		y		y		y			y
Reorder Tone							y							
Resrcs. not Available	y		y				y		y		y			
Silent Monitor		y								y				
Transfer				y		y	y	y	y		y	y		y
Trunks Busy							y				y			
Voice Unit Initiator					y									y

XE " CSTA Event Report - Cause Relationships"§