

libcheckpoint Reference Manual

Generated by Doxygen 1.2.15

Fri Jan 3 16:19:11 2003

Contents

1	libcheckpoint Compound Index	1
1.1	libcheckpoint Compound List	1
2	libcheckpoint File Index	3
2.1	libcheckpoint File List	3
3	libcheckpoint Class Documentation	5
3.1	HC_NodelistNode_t Struct Reference	5
3.2	HC_NodeStack_s Struct Reference	6
3.3	HC_Sequence_t Struct Reference	7
4	libcheckpoint File Documentation	9
4.1	checkpoint.h File Reference	9
4.2	checkpoint_int.h File Reference	15
4.3	HC_Checkpoint.c File Reference	17
4.4	HC_Default.c File Reference	18
4.5	HC_Log.c File Reference	19
4.6	HC_Nodelist.c File Reference	20
4.7	HC_Normal.c File Reference	22
4.8	HC_Rollback.c File Reference	24
4.9	HC_Sequence.c File Reference	26
4.10	HC_Stack.c File Reference	27

Chapter 1

libcheckpoint Compound Index

1.1 libcheckpoint Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

HC_NodelistNode_t	5
HC_NodeStack_s	6
HC_Sequence_t	7

Chapter 2

libcheckpoint File Index

2.1 libcheckpoint File List

Here is a list of all documented files with brief descriptions:

checkpoint.h	9
checkpoint_int.h	15
HC_Checkpoint.c	17
HC_Default.c	18
HC_Log.c	19
HC_Nodelist.c	20
HC_Normal.c	22
HC_Rollback.c	24
HC_Sequence.c	26
HC_Stack.c	27
version.h	??

Chapter 3

libcheckpoint Class Documentation

3.1 HC_NodelistNode_t Struct Reference

```
#include <checkpoint.h>
```

Public Attributes

- **long node_number**
The number corresponding to this node.
- **HC_TypeTag_t type_tag**
One of HC_TypeTag-t.
- **void * data**
Function pointer to the action.

3.1.1 Detailed Description

One node in the Nodelist

The documentation for this struct was generated from the following file:

- **checkpoint.h**
-

3.2 HC_NodeStack_s Struct Reference

```
#include <checkpoint.h>
```

Public Attributes

- **long index**
The node number.
- **HC_NodeStack_s * next**
The next (n-1) node in stack.

3.2.1 Detailed Description

The stack of nodes, used for rolling back

The documentation for this struct was generated from the following file:

- **checkpoint.h**

3.3 HC_Sequence_t Struct Reference

```
#include <checkpoint.h>
```

Public Attributes

- **HC_NodelistNode_t * node_list**
Table of nodes in this sequence.
- **HC_NodeStack_t * index_stack**
Top of the stack.
- long **num_nodes**
Total number of nodes in this sequence.
- long **curr_node**
Current node.
- long **curr_policy_data**
Current policy data (will be passed to policy function).
- **HC_PolicyEvent_t curr_event**
Current event.
- unsigned long **checkpoint**
Last checkpoint value.
- void * **userdata**
User-specific data, passed to all functions.

3.3.1 Detailed Description

Definition of a sequence. This contains everything relating to a specific sequence. It is passed to most functions as context to allow for multiple sequences in an application.

The documentation for this struct was generated from the following file:

- **checkpoint.h**

Chapter 4

libcheckpoint File Documentation

4.1 checkpoint.h File Reference

Compounds

- struct **HC_NodelistNode_t**
- struct **HC_NodeStack_s**
- struct **HC_Sequence_t**

Defines

- #define **HC_OK** 0
Error code: OK.
- #define **HC_FAIL** -1
Error code: Fail.
- #define **HC_INVALIDSEQUENCE** -2
Error code: Invalid Sequence (NULL).
- #define **HC_INVALIDNODE** -3
Error code: Bad node was given.
- #define **HC_FAILEDALLOC** -4
Error code: Out of memory.
- #define **HC_FAILEDSEQUENCE** -5
Error code: ?? unused.
- #define **HC_FINALERROR** -5
Last error value (internal use).

Typedefs

- typedef **HC_NodeStack_s** **HC_NodeStack_t**
-

Enumerations

- enum **HC_PolicyEvent_t** { **HC_INVALID_EVENT**, **HC_NORMALFAIL**, **HC_NORMALSUCCESS**, **HC_ROLLBACKSUCCESS**, **HC_ROLLBACKFAIL** }
- enum **HC_TypeTag_t** { **HC_LISTEND**, **HC_NORMALFUNC**, **HC_ROLLBACKFUNC**, **HC_POLICYFUNC** }

Functions

- int **HC_Normal** (**HC_Sequence_t** *sequence, long node, void *userdata)
- int **HC_NormalCurrent** (**HC_Sequence_t** *sequence, void *userdata)
- int **HC_RollBackPrev** (**HC_Sequence_t** *sequence)
- int **HC_RollBackCurrent** (**HC_Sequence_t** *sequence)
- int **HC_NormalSuccess** (**HC_Sequence_t** *sequence, long policy_data)
- int **HC_NormalFail** (**HC_Sequence_t** *sequence, long policy_data)
- int **HC_RollBackSuccess** (**HC_Sequence_t** *sequence, long policy_data)
- int **HC_RollBackFail** (**HC_Sequence_t** *sequence, long policy_data)
- int **HC_Checkpoint** (**HC_Sequence_t** *sequence, unsigned long checkpoint)
- **HC_Sequence_t** * **HC_NewSequence** (**HC_NodelistNode_t** *nodelist)
- void **HC_DeleteSequence** (**HC_Sequence_t** *sequence)
- int **HC_CallSequence** (**HC_Sequence_t** *sequence, void *userdata, long start_node)
- int **HC_DefaultPolicy** (**HC_Sequence_t** *sequence, void *userdata, **HC_PolicyEvent_t** event, long policy_data)
- void **HC_Panic** (**HC_Sequence_t** *sequence, const char *format,...)
- int **HC_Log** (**HC_Sequence_t** *sequence, const char *format,...)
- const char * **HC_Strerror** (int error)

4.1.1 Detailed Description

This file defines the external interface of the HAFTA checkpoint library.

4.1.2 Typedef Documentation

4.1.2.1 typedef struct **HC_NodeStack_s** **HC_NodeStack_t**

The stack of nodes, used for rolling back

4.1.3 Enumeration Type Documentation

4.1.3.1 enum **HC_PolicyEvent_t**

Event types handled by the policy functions

Enumeration values:

- HC_INVALID_EVENT** An invalid event.
- HC_NORMALFAIL** Normal function failed.
- HC_NORMALSUCCESS** Normal function succeeded.
- HC_ROLLBACKSUCCESS** Rollback function failed.
- HC_ROLLBACKFAIL** Rollback function succeeded.

4.1.3.2 enum HC_TypeTag_t

HC_NodelistNode->type_tag values

Enumeration values:

HC_LISTEND Denotes the end of a Nodelist.

HC_NORMALFUNC Denotes a Normal function on this line.

HC_ROLLBACKFUNC Denotes a Rollback function on this line.

HC_POLICYFUNC Denotes a Policy function on this line.

4.1.4 Function Documentation

4.1.4.1 int HC_CallSequence (HC_Sequence_t * *sequence*, void * *userdata*, long *start_node*)

Execution of a sequence starts here. The sequence starts by running the Normal function for the *start_node* given.

Parameters:

sequence The sequence object, created by **HC_NewSequence()** (p. 26).

userdata User defined data. It is passed to all user functions.

start_node The first executed node.

Returns:

An error code (or HC_OK)

4.1.4.2 int HC_Checkpoint (HC_Sequence_t * *sequence*, unsigned long *checkpoint*)

Sets the checkpoint. This is used during a rollback to determine how far to rollback a function.

Parameters:

sequence Sequence object for context

checkpoint The user-defined checkpoint value

Returns:

An error code.

4.1.4.3 int HC_DefaultPolicy (HC_Sequence_t * *sequence*, void * *userdata*, HC_PolicyEvent_t *event*, long *policy_data*)

Call the policy function for the given node.

Parameters:

sequence Sequence object.

userdata User-defined data.

event The event that has occurred.

policy_data The policy-specific data.

Returns:

An error code.

4.1.4.4 void HC_DeleteSequence (HC_Sequence_t * *sequence*)

Deletes a sequence, cleanups up any memory, etc. The sequence pointer is invalid after this call.

Parameters:

sequence A sequence object

4.1.4.5 int HC_Log (HC_Sequence_t * *sequence*, const char * *format*, ...)

Prints an error to stderr.

Parameters:

sequence Sequence object.

format Printf()-style format string.

4.1.4.6 HC_Sequence_t* HC_NewSequence (HC_NodelistNode_t * *nodelist*)

Creates a sequence object from nodelist.

Parameters:

nodelist A table of nodes and functions to be executed by the sequence.

Returns:

A sequence object containing the nodelist data given.

4.1.4.7 int HC_Normal (HC_Sequence_t * *sequence*, long *node*, void * *userdata*)

Transition forward to a new node.

Parameters:

sequence A sequence object.

node The node to transition to.

userdata User specific data.

Returns:

An error value.

4.1.4.8 int HC_NormalCurrent (HC_Sequence_t * *sequence*, void * *userdata*)

Rerun the normal function for the current node.

Parameters:

sequence A sequence object.

userdata User specific data.

Returns:

An error value.

4.1.4.9 int HC_NormalFail (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a normal function signifying failure.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.1.4.10 int HC_NormalSuccess (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a normal function signifying success.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.1.4.11 void HC_Panic (HC_Sequence_t * *sequence*, const char * *format*, ...)

Called to halt program execution because of a fatal error.

Parameters:

sequence Sequence object.

format Printf()-style format string.

4.1.4.12 int HC_RollBackCurrent (HC_Sequence_t * *sequence*)

Rolls back the current node.

Parameters:

sequence Sequence object.

Returns:

An error code.

4.1.4.13 int HC_RollBackFail (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a rollback function signifying failure in rolling back.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.1.4.14 int HC_RollBackPrev (HC_Sequence_t * *sequence*)

Rolls back the previous node.

Parameters:

sequence Sequence object.

Returns:

An error code.

4.1.4.15 int HC_RollBackSuccess (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a rollback function signifying success in rolling back.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.1.4.16 const char* HC_Strerror (int *error*)

Convert from an error code to a string describing the error.

Parameters:

error The error code

Returns:

A constant string describing the error.

4.2 checkpoint_int.h File Reference

```
#include "checkpoint.h"
```

Functions

- int **HCLPushNode** (**HC_Sequence_t** *sequence, long node)
- long **HCLPopNode** (**HC_Sequence_t** *sequence)
- int **HCLCallNormal** (**HC_Sequence_t** *sequence, long nodenum, void *userdata)
- int **HCLCallRollback** (**HC_Sequence_t** *sequence, long nodenum, void *userdata, unsigned long checkpoint)
- int **HCLCallPolicy** (**HC_Sequence_t** *sequence, long nodenum, void *userdata, **HC_PolicyEvent_t** event, long policy_data)

4.2.1 Detailed Description

Internal data structures and functions for the HAFTA checkpoint library.

4.2.2 Function Documentation

4.2.2.1 int HCLCallNormal (**HC_Sequence_t** * *sequence*, long *nodenum*, void * *userdata*)

Call the normal function for the given node.

Parameters:

- sequence* Sequence object.
nodenum Node number of the normal function.
userdata User-defined data.

Returns:

An error code.

4.2.2.2 int HCLCallPolicy (**HC_Sequence_t** * *sequence*, long *nodenum*, void * *userdata*, **HC_PolicyEvent_t** *event*, long *policy_data*)

Call the policy function for the given node.

Parameters:

- sequence* Sequence object.
nodenum Node number of the policy function.
userdata User-defined data.
event The event that has occurred.
policy_data The policy-specific data.

Returns:

An error code.

4.2.2.3 int HCL_CallRollback (HC_Sequence_t * *sequence*, long *nodenum*, void * *userdata*, unsigned long *checkpoint*)

Call the rollback function for the given node.

Parameters:

sequence Sequence object.

nodenum Node number of the rollback function.

userdata User-defined data.

checkpoint The checkpoint value last cleared in the normal function.

Returns:

An error code.

4.2.2.4 long HCL_PopNode (HC_Sequence_t * *sequence*)

Pops a node from the stack.

Parameters:

sequence The sequence object.

Returns:

The node popped from the stack or an error code (<0)

4.2.2.5 int HCL_PushNode (HC_Sequence_t * *sequence*, long *node*)

Pushes a node onto the stack.

Parameters:

sequence The sequence object.

node The node to be pushed.

Returns:

An error code.

4.3 HC_Checkpoint.c File Reference

```
#include "checkpoint.h"
```

Functions

- `int HC_Checkpoint (HC_Sequence_t *sequence, unsigned long checkpoint)`

4.3.1 Detailed Description

Collection of functions relating to checkpoints.

4.3.2 Function Documentation

4.3.2.1 `int HC_Checkpoint (HC_Sequence_t * sequence, unsigned long checkpoint)`

Sets the checkpoint. This is used during a rollback to determine how far to rollback a function.

Parameters:

sequence Sequence object for context

checkpoint The user-defined checkpoint value

Returns:

An error code.

4.4 HC_Default.c File Reference

```
#include "checkpoint.h"
```

Functions

- int **HC_DefaultPolicy** (**HC_Sequence_t** *sequence, void *userdata, **HC_PolicyEvent_t** event, long policy_data)

4.4.1 Detailed Description

Collection of default functions.

4.4.2 Function Documentation

4.4.2.1 int **HC_DefaultPolicy** (**HC_Sequence_t** * *sequence*, void * *userdata*, **HC_PolicyEvent_t** *event*, long *policy_data*)

Call the policy function for the given node.

Parameters:

- sequence* Sequence object.
- userdata* User-defined data.
- event* The event that has occurred.
- policy_data* The policy-specific data.

Returns:

- An error code.

4.5 HC_Log.c File Reference

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "checkpoint.h"
```

Functions

- void **HC_Panic** (**HC_Sequence_t** *sequence, const char *format,...)
- int **HC_Log** (**HC_Sequence_t** *sequence, const char *format,...)
- const char * **HC_Strerror** (int error)

4.5.1 Detailed Description

Collection of functions relating to logging.

4.5.2 Function Documentation

4.5.2.1 int HC_Log (**HC_Sequence_t** * *sequence*, const char * *format*, ...)

Prints an error to stderr.

Parameters:

- sequence* Sequence object.
format Printf()-style format string.

4.5.2.2 void HC_Panic (**HC_Sequence_t** * *sequence*, const char * *format*, ...)

Called to halt program execution because of a fatal error.

Parameters:

- sequence* Sequence object.
format Printf()-style format string.

4.5.2.3 const char* HC_Strerror (int *error*)

Convert from an error code to a string describing the error.

Parameters:

- error* The error code

Returns:

- A constant string describing the error.

4.6 HC_Nodelist.c File Reference

```
#include "checkpoint_int.h"
```

Functions

- int **HCI_CallNormal** (**HC_Sequence_t** *sequence, long nodenum, void *userdata)
- int **HCI_CallRollback** (**HC_Sequence_t** *sequence, long nodenum, void *userdata, unsigned long checkpoint)
- int **HCI_CallPolicy** (**HC_Sequence_t** *sequence, long nodenum, void *userdata, **HC_PolicyEvent_t** event, long policy_data)

4.6.1 Detailed Description

Collection of functions relating to node lists.

4.6.2 Function Documentation

4.6.2.1 int HCI_CallNormal (**HC_Sequence_t** * *sequence*, long *nodenum*, void * *userdata*)

Call the normal function for the given node.

Parameters:

- sequence* Sequence object.
nodenum Node number of the normal function.
userdata User-defined data.

Returns:

An error code.

4.6.2.2 int HCI_CallPolicy (**HC_Sequence_t** * *sequence*, long *nodenum*, void * *userdata*, **HC_PolicyEvent_t** *event*, long *policy_data*)

Call the policy function for the given node.

Parameters:

- sequence* Sequence object.
nodenum Node number of the policy function.
userdata User-defined data.
event The event that has occurred.
policy_data The policy-specific data.

Returns:

An error code.

4.6.2.3 int HCL_CallRollback (HC_Sequence_t * *sequence*, long *nodenum*, void * *userdata*, unsigned long *checkpoint*)

Call the rollback function for the given node.

Parameters:

sequence Sequence object.

nodenum Node number of the rollback function.

userdata User-defined data.

checkpoint The checkpoint value last cleared in the normal function.

Returns:

An error code.

4.7 HC_Normal.c File Reference

```
#include "checkpoint_int.h"
```

Functions

- **int HC_Normal** (HC_Sequence_t *sequence, long node, void *userdata)
- **int HC_NormalCurrent** (HC_Sequence_t *sequence, void *userdata)
- **int HC_NormalSuccess** (HC_Sequence_t *sequence, long policy_data)
- **int HC_NormalFail** (HC_Sequence_t *sequence, long policy_data)

4.7.1 Detailed Description

Functions relating to the operation of a nodes Normal function.

4.7.2 Function Documentation

4.7.2.1 int HC_Normal (HC_Sequence_t * *sequence*, long *node*, void * *userdata*)

Transition forward to a new node.

Parameters:

- sequence* A sequence object.
- node* The node to transition to.
- userdata* User specific data.

Returns:

An error value.

4.7.2.2 int HC_NormalCurrent (HC_Sequence_t * *sequence*, void * *userdata*)

Rerun the normal function for the current node.

Parameters:

- sequence* A sequence object.
- userdata* User specific data.

Returns:

An error value.

4.7.2.3 int HC_NormalFail (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a normal function signifying failure.

Parameters:

- sequence* Sequence object.
- policy_data* Data specific to the policy

Returns:

An error code.

4.7.2.4 int HC_NormalSuccess (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a normal function signifying success.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.8 HC_Rollback.c File Reference

```
#include "checkpoint_int.h"
```

Functions

- int **HC_RollBackCurrent** (HC_Sequence_t *sequence)
- int **HC_RollBackPrev** (HC_Sequence_t *sequence)
- int **HC_RollBackSuccess** (HC_Sequence_t *sequence, long policy_data)
- int **HC_RollBackFail** (HC_Sequence_t *sequence, long policy_data)

4.8.1 Detailed Description

Collection of functions relating to rolling back nodes in a sequence.

4.8.2 Function Documentation

4.8.2.1 int HC_RollBackCurrent (HC_Sequence_t * *sequence*)

Rolls back the current node.

Parameters:

sequence Sequence object.

Returns:

An error code.

4.8.2.2 int HC_RollBackFail (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a rollback function signifying failure in rolling back.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.8.2.3 int HC_RollBackPrev (HC_Sequence_t * *sequence*)

Rolls back the previous node.

Parameters:

sequence Sequence object.

Returns:

An error code.

4.8.2.4 int HC_RollBackSuccess (HC_Sequence_t * *sequence*, long *policy_data*)

Called from a rollback function signifying success in rolling back.

Parameters:

sequence Sequence object.

policy_data Data specific to the policy

Returns:

An error code.

4.9 HC_Sequence.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include "checkpoint_int.h"
```

Functions

- **int HC_CallSequence** (**HC_Sequence_t** *sequence, void *userdata, long start_node)
- **HC_Sequence_t * HC_NewSequence** (**HC_NodelistNode_t** *nodelist)
- **void HC_DeleteSequence** (**HC_Sequence_t** *sequence)

4.9.1 Detailed Description

Collection of functions relating to sequences.

4.9.2 Function Documentation

4.9.2.1 **int HC_CallSequence** (**HC_Sequence_t** * *sequence*, void * *userdata*, long *start_node*)

Execution of a sequence starts here. The sequence starts by running the Normal function for the start_node given.

Parameters:

- sequence* The sequence object, created by **HC_NewSequence()** (p. 26).
- userdata* User defined data. It is passed to all user functions.
- start_node* The first executed node.

Returns:

An error code (or HC_OK)

4.9.2.2 **void HC_DeleteSequence** (**HC_Sequence_t** * *sequence*)

Deletes a sequence, cleanups up any memory, etc. The sequence pointer is invalid after this call.

Parameters:

- sequence* A sequence object

4.9.2.3 **HC_Sequence_t* HC_NewSequence** (**HC_NodelistNode_t** * *nodelist*)

Creates a sequence object from nodelist.

Parameters:

- nodelist* A table of nodes and functions to be executed by the sequence.

Returns:

A sequence object containing the nodelist data given.

4.10 HC_Stack.c File Reference

```
#include <malloc.h>
#include "checkpoint_int.h"
```

Functions

- int **HCI_PushNode** (HC_Sequence_t *sequence, long node)
- long **HCI_PopNode** (HC_Sequence_t *sequence)

4.10.1 Detailed Description

Collection of functions relating to the node index stack.

4.10.2 Function Documentation

4.10.2.1 long HCI_PopNode (HC_Sequence_t * *sequence*)

Pops a node from the stack.

Parameters:

sequence The sequence object.

Returns:

The node popped from the stack or an error code (<0)

4.10.2.2 int HCI_PushNode (HC_Sequence_t * *sequence*, long *node*)

Pushes a node onto the stack.

Parameters:

sequence The sequence object.

node The node to be pushed.

Returns:

An error code.

Index

checkpoint
 HC_Sequence_t, 7
checkpoint.h, 9
 HC_CallSequence, 11
 HC_Checkpoint, 11
 HC_DefaultPolicy, 11
 HC_DeleteSequence, 11
 HC_FAIL, 9
 HC_FAILEDALLOC, 9
 HC_FAILEDSEQUENCE, 9
 HC_FINALERROR, 9
 HC_INVALID_EVENT, 10
 HC_INVALIDNODE, 9
 HC_INVALIDSEQUENCE, 9
 HC_LISTEND, 11
 HC_Log, 12
 HC_NewSequence, 12
 HC_NodeStack_t, 10
 HC_Normal, 12
 HC_NormalCurrent, 12
 HC_NORMALFAIL, 10
 HC_NormalFail, 12
 HC_NORMALFUNC, 11
 HC_NORMALSUCCESS, 10
 HC_NormalSuccess, 13
 HC_OK, 9
 HC_Panic, 13
 HC_PolicyEvent_t, 10
 HC_POLICYFUNC, 11
 HC_RollBackCurrent, 13
 HC_ROLLBACKFAIL, 10
 HC_RollBackFail, 13
 HC_ROLLBACKFUNC, 11
 HC_RollBackPrev, 13
 HC_ROLLBACKSUCCESS, 10
 HC_RollBackSuccess, 14
 HC_Sterror, 14
 HC_TypeTag_t, 10
checkpoint_int.h, 15
 HCL_CallNormal, 15
 HCL_CallPolicy, 15
 HCL_CallRollback, 15
 HCL_PopNode, 16
 HCL_PushNode, 16

curr_event
 HC_Sequence_t, 7
curr_node
 HC_Sequence_t, 7
curr_policy_data
 HC_Sequence_t, 7

data
 HC_NodelistNode_t, 5

HC_CallSequence
 checkpoint.h, 11
 HC_Sequence.c, 26
HC_Checkpoint
 checkpoint.h, 11
 HC_Checkpoint.c, 17
HC_Checkpoint.c, 17
 HC_Checkpoint, 17
HC_Default.c, 18
 HC_DefaultPolicy, 18
HC_DefaultPolicy
 checkpoint.h, 11
 HC_Default.c, 18
HC_DeleteSequence
 checkpoint.h, 11
 HC_Sequence.c, 26
HC_FAIL
 checkpoint.h, 9
HC_FAILEDALLOC
 checkpoint.h, 9
HC_FAILEDSEQUENCE
 checkpoint.h, 9
HC_FINALERROR
 checkpoint.h, 9
HC_INVALID_EVENT
 checkpoint.h, 10
HC_INVALIDNODE
 checkpoint.h, 9
HC_INVALIDSEQUENCE
 checkpoint.h, 9
HC_LISTEND
 checkpoint.h, 11
HC_Log
 checkpoint.h, 12
 HC_Log.c, 19

- HC_Log.c, 19
 - HC_Log, 19
 - HC_Panic, 19
 - HC_Sterror, 19
- HC_NewSequence
 - checkpoint.h, 12
 - HC_Sequence.c, 26
- HC_Nodelist.c, 20
 - HCI_CallNormal, 20
 - HCI_CallPolicy, 20
 - HCI_CallRollback, 20
- HC_NodelistNode_t
 - data, 5
 - node_number, 5
 - type_tag, 5
- HC_NodelistNode_t, 5
- HC_NodeStack_s
 - index, 6
 - next, 6
- HC_NodeStack_s, 6
- HC_NodeStack_t
 - checkpoint.h, 10
- HC_Normal
 - checkpoint.h, 12
 - HC_Normal.c, 22
- HC_Normal.c, 22
 - HC_Normal, 22
 - HC_NormalCurrent, 22
 - HC_NormalFail, 22
 - HC_NormalSuccess, 22
- HC_NormalCurrent
 - checkpoint.h, 12
 - HC_Normal.c, 22
- HC_NORMALFAIL
 - checkpoint.h, 10
- HC_NormalFail
 - checkpoint.h, 12
 - HC_Normal.c, 22
- HC_NORMALFUNC
 - checkpoint.h, 11
- HC_NORMALSUCCESS
 - checkpoint.h, 10
- HC_NormalSuccess
 - checkpoint.h, 13
 - HC_Normal.c, 22
- HC_OK
 - checkpoint.h, 9
- HC_Panic
 - checkpoint.h, 13
 - HC_Log.c, 19
- HC_PolicyEvent_t
 - checkpoint.h, 10
- HC_POLICYFUNC
 - checkpoint.h, 11
- HC_Rollback.c, 24
 - HC_RollBackCurrent, 24
 - HC_RollBackFail, 24
 - HC_RollBackPrev, 24
 - HC_RollBackSuccess, 24
- HC_RollBackCurrent
 - checkpoint.h, 13
 - HC_Rollback.c, 24
- HC_ROLLBACKFAIL
 - checkpoint.h, 10
- HC_RollBackFail
 - checkpoint.h, 13
 - HC_Rollback.c, 24
- HC_ROLLBACKFUNC
 - checkpoint.h, 11
- HC_RollBackPrev
 - checkpoint.h, 13
 - HC_Rollback.c, 24
- HC_ROLLBACKSUCCESS
 - checkpoint.h, 10
- HC_RollBackSuccess
 - checkpoint.h, 14
 - HC_Rollback.c, 24
- HC_Sequence.c, 26
 - HC_CallSequence, 26
 - HC_DeleteSequence, 26
 - HC_NewSequence, 26
- HC_Sequence_t, 7
 - checkpoint, 7
 - curr_event, 7
 - curr_node, 7
 - curr_policy_data, 7
 - index_stack, 7
 - node_list, 7
 - num_nodes, 7
 - userdata, 7
- HC_Stack.c, 27
 - HCI_PopNode, 27
 - HCI_PushNode, 27
- HC_Sterror
 - checkpoint.h, 14
 - HC_Log.c, 19
- HC_TypeTag_t
 - checkpoint.h, 10
- HCI_CallNormal
 - checkpoint_int.h, 15
 - HC_Nodelist.c, 20
- HCI_CallPolicy
 - checkpoint_int.h, 15
 - HC_Nodelist.c, 20
- HCI_CallRollback
 - checkpoint_int.h, 15
 - HC_Nodelist.c, 20
- HCI_PopNode

- checkpoint_int.h, 16
- HC_Stack.c, 27
- HCI_PushNode
 - checkpoint_int.h, 16
 - HC_Stack.c, 27
- index
 - HC_NodeStack_s, 6
- index_stack
 - HC_Sequence_t, 7
- next
 - HC_NodeStack_s, 6
- node_list
 - HC_Sequence_t, 7
- node_number
 - HC_NodelistNode_t, 5
- num_nodes
 - HC_Sequence_t, 7
- type_tag
 - HC_NodelistNode_t, 5
- userdata
 - HC_Sequence_t, 7