Appendix A: API Summary

Types and Constants

Low level basic types

```
typedef char     CHAR;
typedef unsigned char     UCHAR;
typedef short    SHORT;
typedef unsigned short    USHORT;
typedef long    LONG;
typedef unsigned long    ULONG;
```

Types

```
typedef struct                          CMSession_ *CMSession;
typedef struct                          CMContainer_ *CMContainer;
typedef struct                          CMObject_ *CMObject;
typedef CMObject                        CMProperty;
typedef CMObject                        CMType;
typedef struct                          CMValue_ *CMValue;
typedef CHAR                            *CMOpenMode;
typedef CHAR                            *CMGlobalName;
typedef CHAR                            *CMErrorString;
typedef CM_CHAR                          *CMMetaData;
typedef void                            *CMRefCon;
typedef void                             *CMPtr;
typedef UCHAR                           *CMMagicBytes;
typedef CM_UCHAR                        *CMDataPacket;
typedef CM_UCHAR                        *CMDataBuffer;
typedef CM_UCHAR                        *CMPrivateData;
typedef CM_UCHAR                        CMReference[4];
typedef UCHAR                           CMSeekMode;
typedef UCHAR                           CMBoolean;
typedef USHORT                          CMContainerUseMode;
typedef USHORT                          CMContainerFlags;
typedef CM_USHORT                    CMContainerModeFlags;
typedef USHORT                       CMEofStatus;
typedef CM_LONG                      CMErrorNbr;
typedef ULONG                        CMGeneration;
typedef ULONG                        CMSize;
typedef ULONG                        CMCount;
typedef void                         *CMPtr;
typedef void                         ( *CMHandlerAddr)();
typedef  CMHandlerAddr ( *CMMetaHandler)
                     (CMType,
                     const CMGlobalName);
```

Constants

```
const CMContainerUseMode             kCMReading = 0x0001
const  CMContainerUseMode            kCMWriting = 0x0002
const CMContainerUseMode             kCMReuseFreeSpace = 0x0004
const CMContainerUseMode             kCMUpdateByAppend = 0x0008
const CMContainerUseMode             kCMUpdateTarget = 0x0010
const CMContainerUseMode             kCMConverting = 0x0020;
const CMSeekMode                     kCMSeekSet = 0x00;
const CMSeekMode                     kCMSeekCurrent= 0x01;
const CMSeekMode                     kCMSeekEnd = 0x02;
```

Operation Definitions

Session Operations

```
CMSession  CMStartSession(CMMetaHandler metaHandler,
                     CMRefCon sessionRefCon)
void  CMEndSession(CMSession sessionData,
                     CMBoolean closeOpenContainers)
void  CMAbortSession(CMSession sessionData);
CMRefCon  CMGetSessionRefCon(CMContainer container)
void  CMSetSessionRefCon(CMContainer container,
                     CMRefCon refCon)
```

```
CMHandlerAddr  CMSetMetaHandler(const CMSession sessionData,
                     const CMGlobalName typeName,
                     CMMetaHandler metaHandler)

CMHandlerAddr  CMGetMetaHandler(const CMSession sessionData,
                     const CMGlobalName typeName)

CMHandlerAddr CMGetOperation(CMType targetType,
                     const CMGlobalName operationType);
```

Container Operations

```
CMContainer  CMOpenContainer(CMSession sessionData,
                     CMRefCon attributes,
                     const CMGlobalName typeName,
                     CMContainerUseMode useFlags);

CMContainer  CMOpenNewContainer(CMSession sessionData,
                     CMRefCon attributes,
                     const CMGlobalName typeName,
                     CMContainerUseMode useFlags,
                     CMGeneration generation,
                     CMContainerFlags containerFlags,
                     ...);

void CMGetContainerInfo(const CMContainer container,
                     CMGeneration *generation,
                     CMContainerFlags *containerFlags,
                     CMGlobalName typeName);

CMSession  CMGetSession(CMContainer container)

VOID CMCloseContainer(CMContainer container);

VOID CMAbortContainer(CMconst_CMContainer container);
```

Type and Property Operations

```
CMType CMRegisterType(CMContainer targetContainer,
                     const CMGlobalName name);

CMProperty CMRegisterProperty(CMContainer targetContainer,
                     const CMGlobalName name);

CMBoolean CMIsType(CMObject theObject);

CMBoolean CMIsProperty(CMObject theObject);

CMType CMGetNextType(CMContainer targetContainer,
                     CM CMType currType);

CMProperty CMGetNextProperty(CMContainer targetContainer,
                     CMProperty currProperty);
```

```
CMCount  CMAddBaseType(CMType type,
                    CMType baseType)

CMCount  CMRemoveBaseType(CMType type,
                    CMType baseType)
```

Object Operations

```
CMObject CMNewObject(CMContainer targetContainer);

CMObject CMGetNextObject(CMContainer targetContainer,
                    CMObject currObject);

CMProperty CMGetNextObjectProperty(CMObject theObject,
                    CMProperty currProperty);

CMObject CMGetNextObjectWithProperty(CMContainer targetContainer,
                    CMObject currObject,
                    CMProperty property)

CMContainer CMGetObjectContainer(CMObject theObject);

CMGlobalName CMGetGlobalName(CMObject theObject);

CMRefCon  CMGetObjectRefCon(CMObject theObject)

void  CMSetObjectRefCon(CMObject theObject,
                    CMRefCon refCon)

VOID CMDeleteObject(CMObject theObject);

VOID CMDeleteObjectProperty(CMObject theObject,
                    CMProperty theProperty);

VOID CMReleaseObject(CMObject theObject);
```

Value Operations

```
CMCount CMCountValues(CMObject object,
                    CMProperty property,
                    CMType type);

CMValue CMUseValue(CMObject object,
                    CMProperty property,
                    CMType type);

CMValue  CMGetNextValue(CMObject object,
                    CMProperty property,
                    CMValue currValue)

CMValue CMNewValue(CMObject object,
                    CMProperty property,
                    CMType type,
                    ...);
```

```
CMValue  CMVNewValue(CMObject object,
                    CMProperty property,
                    CMType type,
                    va_list dataInitParams)

CMSize CMGetValueSize(CMValue value);

CMSize CMReadValueData(CMValue value,
                    CMPtr buffer,
                    CMCount offset,
                    CMSize maxSize)

void CMWriteValueData(CMValue value,
                    CMPtr buffer,
                    CMCount offset,
                    CMSize size)

VOID CMInsertValueData(CMValue value,
                    CMPtr buffer,
                    CMCount offset,
                    CMSize size)

VOID CMDeleteValueData(CMValue value,
                    CMCount offset,
                    CMSize size)

VOID CMDefineValueData(CMValue value,
                    CMSize offset,
                    CMSize size);

void  CMMoveValue(CMValue value,
                    CMObject object,
                    CMProperty property)

VOID CMGetValueInfo(CMValue value,
                    CMContainer *container,
                    CMObject *object,
                    CMProperty *property,
                    CMGeneration *generation);

void  CMSetValueType(CMValue value,
                    CMType type)

void  CMSetValueGeneration(CMValue value,
                    CMGeneration generation)

void CMDeleteValue(CMValue value);

void CMReleaseValue(CMValue value);
```

Reference Operations

```
CMReference   *CMNewReference(CMValue value,
                    CMObject referencedObject,
                    CMReference theReferenceData)

CMObject  CMGetReferencedObject(CMValue value,
                    CMReference theReferenceData)

CMReference  *CMSetReference(CMValue value,
                    CMObject referencedObject,
                    CMReference theReferenceData);

void CMDeleteReference(CMValue value,
                    CMReference theReferenceData);

CMCount CMCountReferences(CMValue value);

CMReference *CMGetNextReference(CMValue value,
                    CMReference currReferenceData);
```