```
/*
** RELEASE STATEMENT(s):
**                          UNLIMITED RIGHTS
**  The Government has the right to use, modify, reproduce, release, perform,
**  display, or disclose this application programmable interface in whole or in
**  part, in any manner and for any purpose whatsoever, and to have or
**  authorize others to do so.
**
** Distribution Statement A - Approved for public release; distribution is
** unlimited (27 August 2015).
*/

/*
** JTNC Standard:
**  Software Communications Architecture
**  Appendix C: Core Framework Interface Description Language (IDL)
**  Version: 4.1, 20 August 2015
*/

//Source file: CFCommonTypes.idl

#ifndef __CFCOMMONTYPES_DEFINED
#define __CFCOMMONTYPES_DEFINED

#include "CFPrimitiveTypes.idl"
#include "CFPrimitiveSeqTypes.idl"
#include "CFProperties.idl"

module CF {

   /* This type is an unbounded sequence of octets. */
   typedef CF::OctetSeq OctetSequence;

   /* This type defines a sequence of strings. */
   typedef sequence <string> StringSequence;

   /* This enum is used to pass error number information in various
      exceptions. Those exceptions starting with "CF_E" map to the POSIX
      definitions.
      The "CF_" has been added to the POSIX exceptions to avoid namespace
      conflicts.  CF_NOTSET is not defined in the POSIX specification.
      CF_NOTSET is an SCA specific value that is applicable for any
      exception when the method specific or standard POSIX error values
      are not appropriate. */
   enum ErrorNumberType {
      CF_NOTSET,
      CF_E2BIG,
      CF_EACCES,
      CF_EAGAIN,
      CF_EBADF,
      CF_EBADMSG,
      CF_EBUSY,
      CF_ECANCELED,
      CF_ECHILD,
      CF_EDEADLK,
      CF_EDOM,
      CF_EEXIST,
      CF_EFAULT,
      CF_EFBIG,
      CF_EINPROGRESS,
      CF_EINTR,
      CF_EINVAL,
      CF_EIO,
      CF_EISDIR,
      CF_EMFILE,
      CF_EMLINK,
```

```
        CF_EMSGSIZE,
        CF_ENAMETOOLONG,
        CF_ENFILE,
        CF_ENODEV,
        CF_ENOENT,
        CF_ENOEXEC,
        CF_ENOLCK,
        CF_ENOMEM,
        CF_ENOSPC,
        CF_ENOSYS,
        CF_ENOTDIR,
        CF_ENOTEMPTY,
        CF_ENOTSUP,
        CF_ENOTTY,
        CF_ENXIO,
        CF_EPERM,
        CF_EPIPE,
        CF_ERANGE,
        CF_EROFS,
        CF_ESPIPE,
        CF_ESRCH,
        CF_ETIMEDOUT,
        CF_EXDEV
    };

    /* This exception indicates an invalid file name was passed
       to a file service operation. The message provides information
       describing why the filename was invalid. */
    exception InvalidFileName {
        CF::ErrorNumberType errorNumber;
        string msg;
    };

    /* This exception indicates an invalid object reference error. */
    exception InvalidObjectReference {
        string msg;
    };

    /* This structure defines a port. */
    struct PortAccessType {
        string portName;
        Object portReference;
    };

    /* This type defines an name/value sequence of PortAccessType
       structures. */
    typedef sequence <PortAccessType> Ports;

    /* This enumeration defines the basic component types. */
    enum ComponentEnumType {
        APPLICATION_COMPONENT,
        MANAGEABLE_APPLICATION_COMPONENT,
        DEVICE_COMPONENT,
        LOADABLE_DEVICE_COMPONENT,
        EXECUTABLE_DEVICE_COMPONENT,
        MANAGEABLE_SERVICE_COMPONENT,
        SERVICE_COMPONENT,
        DEVICE_MANAGER_COMPONENT,
        DOMAIN_MANAGER_COMPONENT,
        APPLICATION_MANAGER_COMPONENT,
        APPLICATION_FACTORY_COMPONENT,
        APPLICATION_COMPONENT_FACTORY_COMPONENT,
        PLATFORM_COMPONENT_FACTORY_COMPONENT
    };

    /* This structure defines the basic elements of a component. */
```

```
   struct ComponentType {
      string identifier;
      string profile;
      CF::ComponentEnumType type;
      Object componentObject;
      CF::Ports providesPorts;
      CF::Properties specializedInfo;
   };

   /* This type defines an unbounded sequence of objects. */
   typedef sequence <Object> ObjectSequence;

};
#endif
```