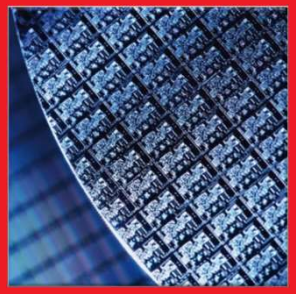
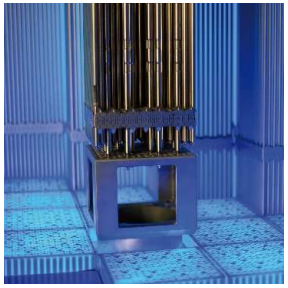
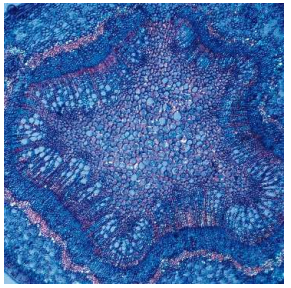
CAMECA
SCIENCE & METROLOGY SOLUTIONS







NFDI Nexus Workshop

March 2022

AMETEK®
MATERIALS ANALYSIS DIVISION

Open Data Access Initiatives

- CAMECA's AP Suite software works with an open container format file meant to facilitate access to
  - *The .APT file contains a number of sections each with a distinct type of data relevant for a particular analysis*
  - *It provides an extensible format which will accommodate new types of data classifications, derived data, or meta data, in the future without changing the format*
  - *Created as a lightweight format to enable custom analyses*
  - *Full specification is in AP Suite 6 User Guide appendix C*
- In parallel, we are monitoring efforts of the IFES society to define an HDF5 format and hashing protocol with the intention of providing support once adopted

File Header

Every .APT file starts with a header with the following information:

```
struct APTFileHeader
{
    char        cSignature[4];          // "APT\0"
    int         iHeaderSize;            // Header size, bytes (540)
    int         iHeaderVersion;         // Version # (2)
    wchar_t     wcFilename[256];        // Original filename, UTF16, null terminated.
    FILETIME    ftCreationTime;         // Original file creation time
    int64       llIonCount;             // Number of ions represented by file
};
```

Section Header

Following the file header, there can be any number of sections. Each section starts with the following header:

```
struct APTSectionHeader
{
    char                cSignature[4];      // Section signature, "SEC\0"
    int                 iHeaderSize;        // Size of the section header, bytes (148)
    int                 iHeaderVersion;     // Version # of this section header (2)

    wchar_t             wcSectionType[32];  // String representation of the section type.  UTF16
    int                 iSectionVersion;    // Version of this section data
    REL_TYPE            eRelationshipType;  // How the records relate to ion #

    RECORD_TYPE         eRecordType;        // Type of record (fixed, variable, etc.)
    RECORD_DATATYPE     eRecordDataType;    // The data type of the records (int, float, etc.)
    int                 iDataTypeSize;      // The size, in bits, of the data type (eg 8 = 1 byte)
    int                 iRecordSize;        // Size of the record (bytes). This must be a multiple
                                            //   of iDataTypeSize and 8, or 0 for variable length
    wchar_t             wcDataUnit[16];     // UTF16 string representing the unit of the data (eg "nm")
    int64               llRecordCount;      // Number of records following this header.
                                            //   DO NOT USE for seeking to next section, use llByteCount
    int64               llByteCount;        // Number of bytes following the header.  This may be
                                            //   > llRecordCount * iRecordSize to allow for padding
};
```