Tip of the Week :

- How to use tagged structures and bitfields

Closer Look at DTYPE

```
typedef struct
{
  UINT32 dArrayLength;
  short  dFormat;
  short  xferReason;
  int    dStamp;
  int    sysStamp;
  double dTimeStamp;
  char   dTag[TAG_NAME_SIZE];
  DUNION data;
} DTYPE;
```

```
typedef union
{
  void   *vptr;
  char   *cptr;
  UINT8  *bptr;
  SINT16 *sptr;
  UINT16 *usptr;
  SINT32 *lptr;
  UINT32 *ulptr;
# ifdef SINT64
  SINT64 *llptr;
# endif
# ifdef UINT64
  UINT64 *ullptr;
# endif
  float  *fptr;
  double *dptr;
} DUNION;
```

dTag used to identify structures and bitfields if the format is CF_STRUCT or CF_BITFIELD

Note: In the Java API

- TDataType has dTag field and getTag() method.
- Tag supplied with the TTaggedStructure or TBitfield constructor.

Tagged Structures

- User-defined types
  - Not systematically known (by definition)
  - Must register themselves with the local structure registry.
    - AddFieldToSruct(tag,addr,num,type,field)
    - SealTaggedStruct(tag,structsize,maxarraysize);

Tagged Structures (example)

Structure registration :

```
typedef struct
{
  float amplitude;
  float frequency;
  float noise;
  float phase;
  int   numberCalls;
  char  description[64];
} SineInfo;

#define quit(i) { printf("Register struct: out of memory\n"); exit(i); }
void registerStructs(void)
{
  static int done = 0;
  if (done) return;
  done = TRUE;
  /* this must follow the order of the structure explicitly! */
  if (addFieldToStruct("SineInfo",OFFSETIN(SineInfo,amplitude),1,CF_FLOAT,"amplitude")) quit(1);
  if (addFieldToStruct("SineInfo",OFFSETIN(SineInfo,frequency),1,CF_FLOAT,"frequency")) quit(1);
  if (addFieldToStruct("SineInfo",OFFSETIN(SineInfo,noise),1,CF_FLOAT,"noise")) quit(1);
  if (addFieldToStruct("SineInfo",OFFSETIN(SineInfo,phase),1,CF_FLOAT,"phase")) quit(1);
  if (addFieldToStruct("SineInfo",OFFSETIN(SineInfo,numberCalls),1,CF_LONG,"numberCalls")) quit(1);
  if (addFieldToStruct("SineInfo",OFFSETIN(SineInfo,description),64,CF_TEXT,"description")) quit(1);
  /* terminate the structure definition like this! */
  if (sealTaggedStruct("SineInfo",sizeof(SineInfo),NUM_DEVICES)) quit(1);

  /* below a status header struct */
  if (addFieldToStruct("StHdr",OFFSETIN(StHdr,a),1,CF_INT32,"a")) quit(1);
  if (addFieldToStruct("StHdr",OFFSETIN(StHdr,b),1,CF_FLOAT,"b")) quit(1);
  if (addFieldToStruct("StHdr",OFFSETIN(StHdr,t),16,CF_TEXT,"t")) quit(1);
  if (sealTaggedStruct("StHdr",sizeof(StHdr),NUM_DEVICES)) quit(1);

  /* below a status body struct */
  if (addFieldToStruct("StBod",OFFSETIN(StBod,c),1,CF_INT32,"c")) quit(1);
  if (addFieldToStruct("StBod",OFFSETIN(StBod,d),1,CF_FLOAT,"d")) quit(1);
  if (addFieldToStruct("StBod",OFFSETIN(StBod,e),1,CF_DOUBLE,"e")) quit(1);
  if (sealTaggedStruct("StBod",sizeof(StBod),NUM_DEVICES)) quit(1);

  /* below a struct composed of the above header a 4 X the above body : */
  if (addFieldToStruct("StCmp",OFFSETIN(StCmp,hdr),1,CF_STRUCT,"<StHdr>hdr")) quit(1);
  if (addFieldToStruct("StCmp",OFFSETIN(StCmp,body),4,CF_STRUCT,"<StBod>body")) quit(1);
  if (sealTaggedStruct("StCmp",sizeof(StCmp),NUM_DEVICES)) quit(1);
```

Property Registration

Per API:

```
dout.dFormat = CF_STRUCT;
strncpy(dout.dTag,"SineInfo",TAG_NAME_SIZE);
din.dFormat = CF_STRUCT;
strncpy(din.dTag,"SineInfo",TAG_NAME_SIZE);
RegisterPropertyInformation(SINEQM_TAG,"SineInfo",&dout,&din,CA_READ,AT_UNKNOWN,
  10,"Sine Generator Information",PRP_INFO,NULL);
```

Per exports.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	CONTEXT	EXPORT_NAME	LOCAL_NAI	PROPERTY	PROPERT	PROPERT	ACCESS	FORMAT	NUM_MOI	DESCRIPT	REDIRECTION		
2	TEST	WinSineServer	SINEQM	Sine	8192	1	READ.SPECTF	float	10	[-1000:1000 V][0:1000 ms]Sine Curve			
3	TEST	WinSineServer	SINEQM	Amplitude	10	2	READ|WRITE	float.CHANNEL	10	[1:1000 V]Sine Curve Amplitude			
4	TEST	WinSineServer	SINEQM	Frequency	10	3	READ|WRITE	float.CHANNEL	10	[1:60]Sine Curve Frequency			
5	TEST	WinSineServer	SINEQM	Phase	10	4	READ|WRITE	float.CHANNEL	10	[0:512]Sine Curve Phase			
6	TEST	WinSineServer	SINEQM	Noise	10	5	READ|WRITE	float.CHANNEL	10	[0:100 V]Sine Curve Noise Level			
7	TEST	WinSineServer	SINEQM	SineInfo	10	6	READ|WRITE	struct.SineInfo	10	Sine Generator Information			
8	TEST	WinSineServer	SINEQM	Status	10	7	READ	BITFIELD16.StsBits	10	Status bits			
9	TEST	WinSineServer	SINEQM	StructTest	10	8	READ|WRITE	struct.StCmp	10	struct test			
10	TEST	WinSineServer	SINEQM	SpectrumTest	8192	9	READ|WRITE	spectrum	10	Spectrum test			
11	TEST	WinSineServer	SINEQM	ImageTest	8192	10	READ	image	10	Image test			
12													

Property Registration

Per fec.xml :

```
      <REDIRECTION />
    </PROPERTY>
  - <PROPERTY>
      <NAME>SineInfo</NAME>
      <DEVICE_SET />
      <EGU />
      <MAX />
      <MIN />
      <ID>4</ID>
      <DESCRIPTION>Sine Curve Info Structure</DESCRIPTION>
      <SIZE_IN>1</SIZE_IN>
      <DTYPE_IN>struct.SineInfo</DTYPE_IN>
      <SIZE_OUT>1</SIZE_OUT>
      <DTYPE_OUT>struct.SineInfo</DTYPE_OUT>
      <ACCESS>READ|WRITE</ACCESS>
      <REDIRECTION />
    </PROPERTY>
  </EOM>
```

Per Java API :

```
public TExportProperty(int prpId, String prpName, String prpDescription, int prpSizeOut,
    short prpFormatOut, String prpTagOut, int prpSizeIn, short prpFormatIn, String prpTagIn)
{
```

Structure Registration
(client-side)

- client can do the same as the server
  - is there is logic which needs to use the fields for something, this must happen !
  - Client and server programmer usually the same person or team.
- client can discover the struct
  - works for display (e.g. Instant Client).

Tagged Structures
(client-side)

- Calls use dFormat = CF_STRUCT, fill in the dTag and provide a reference pointer to the structure (that's all).
- Byte swapping, alignment handled 'underneath'.
- Possible Issue:
  - no 'server' reference in the registry
  - possible problem if a client talks to server A with his version of struct "MYSTRUCT" and server B with another version of "MYSTRUCT".
  - BUT: usually systematically defined and used.
  - has never surfaced as a problem!
    - TINE specific structs "PRPQSr4", "CLNQsr4", etc.
    - Server specific structs "DHS" from Event Server
    - Subsystem specific structs "TRCHDR" for Transient Recorders (RF)

Bitfields

- Also user-defined ‘types’
  - Not systematically known (by definition)
  - Must register themselves with the local structure registry.
    - OpenBitField(srv,tag,format)
    - AddFieldToBitField(srv,tag,mask,field)

Bitfield (example)

Register the bitfield (analogous to struct)

```
openBitField(SRVTAG,"StsBits",CF_BITFIELD16);
addFieldToBitField(SRVTAG,"StsBits",0x01,"field1");
addFieldToBitField(SRVTAG,"StsBits",0x02,"field2");
addFieldToBitField(SRVTAG,"StsBits",0x04,"field3");
addFieldToBitField(SRVTAG,"StsBits",0x08,"field4");
addFieldToBitField(SRVTAG,"StsBits",0xf0,"field5");
addFieldToBitField(SRVTAG,"StsBits",0xf00,"field6");
addFieldToBitField(SRVTAG,"StsBits",0xf000,"field7");
```

Register the property :

```
dout.dFormat = CF_BITFIELD16;
strncpy(dout.dTag,"StsBits",TAG_NAME_SIZE);
RegisterPropertyInformation(SINEQM_TAG,"SineStatus",&dout,&din,CA_READ,AT_UNKNOWN,
  10,"Sine Generator Status",PRP_STATUS,NULL);
```

Bitfields
(client-side)

- client can do the same as the server
  - is there is logic which needs to use the fields for something, this must happen !
  - Client and server programmer usually the same person or team.
- client can discover the bitfield
  - works for display (e.g. Instant Client – but don’t look for now).

Bitfields
(client-side)

- Never systematically defined/known
- CDI database A will most likely be independent of database B
- clients will probably need to access multiple CDI databases (e.g. instant client)
- There is a 'server' reference in the bitfield registry!
- no problem with server A using bitfield "StatusBits" and server B using the same name.

- No problem with name collisions !

Advantages:

- Structures:
  - A collection of information that needs to be handled atomically !
- Bitfields:
  - Bits or groups of bits can be named and addressed (read not write!)
  - Communication client-server ALWAYS deals with the whole bitfield.
    - N calls for N different fields from the same bitfield result in one and only one contract client-server.