Structures, Databases and Devices

Structures and databases

D2000 system defines the term **structure** as a particular number of named items, to which it is possible to specify the properties as a value type, start value, saving start value, limits. Structure in the system is defined by an object Structure definition, so a specific name is assigned to it. Structure definition (holds no value) represents a particular model, which is used (in the way of a reference) in the definition of other objects:

- Structured variable
- Database

An object of Structured variable type is closely associated with an object of Structure definition type. It defines a non-zero number of rows of values. The structure of each row is given by an object of Structure definition type. A value matrix is created:

Example:
The object SD.PersonDef of Structure definition contains the following items:

Item name	Item type
Name	Text
Age	Integer
Born	Absolute time
Children	Integer

The object SV.Persons is a Structured variable type and contains five (5) rows. The rows of this object are defined by the object SD.PersonDef. Object value includes twenty (20) values [5 x 4] of various types, the arrangement of which is shown in the following table:

Row/Column	Name	Age	Born	Children
1				
2				
3				
4				
5				

Each item represents one value, for which there are defined (and uniquely for each value) all usual properties in the D2000 system (start value, limits, status bits, etc.). Individual values can be displayed in pictures, used in expressions (SV.Persons[3]^Age), in eval tags, or events.

Using an object of Database type, it is possible to access a SQL database (via the interface ODBC), structure of which (names and types of columns) corresponds to an object of Structure definition type.

For each column (item) at the level of an object of Structure definition it is possible to define the following attributes:

- Name*
- Description*
- Value type*
- Status text (for displaying)
- Limits
- Start value
- Index of transformation palette

These properties are used (or they are not, according to configuration) for columns in the objects of Structured variable and *Database* types. For objects of Structured variable type, these properties can be configured for each item (value) individually.

* Attribute is defined at the level of the object of Structure definition type and cannot be changed.

Devices

In some cases, there are certain disadvantages to using structures:

- **Possibility of duplicate connection of objects:** connection of one object into several rows of a structured variable (for the value type Object).
- **Anonymous rows**: individual rows of the structure are addressed by a numeric index. If the rows of one structured variable belong to several groups (e.g. each row corresponds to one generator and several generators create a block and several blocks create a power plant), this is not obvious when working with numerical indices and a mistake may occur.
- **Memory fragmentation**: structured variables are represented in the D2000 system as contiguous blocks of memory. When using large structures (tens to hundreds of rows and columns), memory fragmentation occurs (especially in the D2000 Server process).
- **Archiving problems**: when using structured archives (archiving a column or possibly an entire structured variable), all values are stored in one database table, which can be large and its maintenance (reorganization of data and indexes) can be demanding in terms of disk operations and CPU. At the same time, such a structured archive object is assigned to one write archive task, so that writes cannot be parallelized. If other archive objects (computed and/or statistical archives) are built on top of such an archive object, each of them is again handled by one write archive task. If e.g. a recalc of the archive by the RECALC tell command is started, it is serialized and performed sequentially for individual rows of the structured archive.
- **Editing**: a structured variable can only be edited by one user at a time.

These problems are solved by the device concept brought by the D2000 in version 21. The **Device** can be perceived as a one-line structured variable, whereby:

- The Device definition object defines the structure of the Device type object - it is a *Structured device definition*.
- The Device definition supports all types of columns such as Structure definition, in addition, columns of **Internal item** type (used to connect another D2000 object to the device) and **Device** (used to connect another device - subdevice).
- The Device definition can alternatively be defined as an array of Devices that are of another *Device Definition* type - this is an *Array device definition* (e.g. let's have DD.Generator and DD.GeneratorArr derived from it).
- It is thus possible to build a "tree" of devices, while the D2000 Server process ensures a consistent naming convention for objects connected to the device items of **Internal item** and **Device type**.

Thus, in principle, the concept of devices allows the creation of **tree structures** with the provision of a uniform naming convention of the objects from which the trees are created.

For each column (item) at the level of an object of Device definition it is possible to define the following attributes:

- Name *
- Description *
- Value type *
- Status text (for displaying)
- Limits
- Start value
- Index of transformation palette
- Device type *
- Mandatory *

These properties are used (or they are not, according to configuration) for columns in the objects of Device types. For objects of Device type, these properties can be configured for each item individually (with the exception of attributes marked by an asterisk).

* Attribute is defined at the level of the object of Device definition type and cannot be changed.

You can create template schemes, template eval tags, and template archives for devices. All of these object types are configured using a Device definition. Using the template pictures, the Device with the corresponding Device definition can then be displayed. Configuring template eval tags and template archives will cause instances of these objects, i.e. eval tags and archived objects for each Device with the relevant Device definition, to be created.