ReqView File Data Format version 0.11

This document describes internal data format of ReqView project files. ReqView project files have ".reqw" extension and "application/vnd.eccam.reqview+json" MIME type. ReqView files have JSON format and can be stored human readable or encrypted.

ReqView Project File

Each ReqView project file has following header:

```
{
  id:                "ReqView",
  // Metadata
  metadata: {
    format:          "0.11",

    // File encryption
    cipher:          "aes-256-cbc",                      // for encrypted file only
    iv:              "bc55b87f981c3c0988fab4abfd8eaeeb", // for encrypted file only
    salt:            "nohz0Ahm5"                         // for encrypted file only
  },
  // Content
  documents:         [ /* Documents */ ],
  attachments:       [ /* Attachments */ ],
  traceability:      { /* Traceability */ }
}
```

Where:

- *id* is unique identifier of a project
- *metadata*:
  - *format* is the data format version string
  - *cipher* is OpenSSL cipher name, currently we support only aes-256 with CBC mode
  - *iv* is an initialization vector for the CBC mode, it is computed by OpenSSL when creating the key from user supplied password
  - *salt* is needed for decryption because javascript function does not generate the *iv* same as OpenSSL
- *documents* is list of documents described in Documents section, if file is encrypted then documents are encoded as base64 text without line feeds
- *attachments* is a list of external attachments (images, documents, ...) described in Attachments section; if file is encrypted then attachments are encoded as base64 text without line feeds
- *traceability* describes configuration of links between objects.

Documents

```
[
  { // Embedded document
    id:                 "SR",
    name:               "Document name",
    createdOn:          "2013-10-10T22:58:09.370Z",
    lastId:             123,
    attributes:         { ... },
    data:               [ ... ],
    view:               { ... },              // optional
    synchronizer: {                           // optional, filled by Synchronizer only
      mapping:  [
        { attribute: "id",          external: "Object Identifier"},
        { attribute: "section",     external: "Object Number"},
        { attribute: "heading",     external: "Object Heading"},
        { attribute: "text",        external: "Object Text"},
        { attribute: "level",       external: "Object Level"},
        { attribute: "discussion",  external: "ReqView_Discussion"},
        { attribute: "history",     external: "ReqView_History"},
        { attribute: "status",      external: "ReqView_Status"}
      ]
    }
  },
  { // External document stored in an separate file (not yet implemented)
    id:                 "UR",
    file:               "UR.reqw-doc"
  },
  ...
]
```

Where:

- *id* is unique document identifier in the project
- *name* is document name string
- *createdOn* is file creation date in ISO format
- *lastId* is ID of the last created object (integer >=0)
- *attributes* describes attributes configuration, see Attributes Configuration (RequirementAttributes.html#Attributes)
- *data* is array of objects (requirements, tests, etc.) described in section Data
- *view* describes visual configuration of columns in the Table View, see section View
- *synchronizer* stores properties which Synchronizer uses for pairing ReqView document with its original import from a RM tool
    - *mapping* describes mapping of ReqView attributes to the original RM tool attributes
- *file* stores file path to an external document

Document can be also stored in a separate file. In that case *file* document property is file name storing the external document.

Data

Data section stores array of objects corresponding to the highest level sections. Each section has property *children* storing array of children objects.

```
[
  { // Section 1.
    // Attributes
    id:           "27",
    heading:      "Section 1. Name",       // only for section
    status:       "Draft",
    discussion: [
      { // Comment
        comment: "Some comment"
        date: "2012-11-21T15:37:59.099Z",
        author: { name: "Libor Bus", email: "libor.bus@eccam.com", company: "Eccam" }
      },
      { /* Another comment */ }, ...
    ],
    history: [
      { // Change record
        <property>: <orig_value>,
        date: "2012-11-21T15:37:59.099Z",
        author: { name: "Libor Bus", email: "libor.bus@eccam.com", company: "Eccam" }
      },
      { /* Another change record */ }, ...
    ],
    attachments: [
      { id: "UR-27_0_filename.ext" },
      { /* Another attachment */ }, ...
    ]
    children: [
      { // Requirement 1.0-1
        // Attributes
        id:           "38",
        text:         "Requirement text",   // only for requirement
        status:       "Approved",
        discussion:   [ ... ],
        history:      [ ... ],
        links:        {
            linkTypeId: [ "25", ... ]
        }
      },
      { /* Another requirement */ }, ...
    ]
  },
  { /* Another section */ }, ...
]
```

Where:

- *id* is an unique integer identifier
- *heading* is a section heading string
- *text* is a object text description
- *discussion* is an array of comments where each comment stores date (ISO string), author name, email and company name and comment text

- *status* is custom attribute defined in Section Attributes Configuration.
- *history* is array of change records where each change record stores changed property, date of change (ISO string) and author of change
- *attachments* is array of references to attachments, see Section Attachments
- *children* is an ordered array of direct children (both sections and requirements)
- *links* is an object storing links. The example illustrates an outgoing *Reference* link to the object with *id 25* in the same document, see Section Traceability

View

View object stores definition of visible columns in the Table View:

```
{
  columns: [
    {
      attribute:    "id",
      width:        "100px",   // optional
    },
    {
      attribute:    "description",
    },
    {
      attribute:    "status",
      hidden:       true         // optional
    }
    ...
  ]
}
```

Where:
- *columns* is ordered array of columns, where each column can have one of following properties:
  - *attribute* an identifier of attribute, i.e. an internal attribute "id", "description", "discussion" or a custom attribute, e.g. "status",
  - *width* is optional width of the column, e.g. "100px", "20em", "30%" or "auto" (column will use whatever space remains). Default value is "auto".
  - *hidden* is optional flag which specifies if the column is hidden. Default value is false.

If View section or *columns* definition is omitted then all attributes defined in section Attributes are displayed in the columns in alphabetical order. If View section does not specify all the attributes then columns for the missing attributes are placed behind the attributes specified in View section in alphabetical order.

Attachments

```
[
  { // Embedded attachment
    id:     "UR-27_0_filename.ext",
    data:   "data:image/png;base64,iVBORw0KGg ... ",
  },
  { // External attachment (not yet implemented)
    id:     "UR-27_1_filename.ext",
    type:   "image/png",
    file:   "attachments/27_1_filename.ext"
  },
  ...
]
```

Where:

- *id* is an unique attachment indentifier in the following form: *reqid_seqnum_filename.ext*, *reqid* is *id* of the related object, *seqnum* is sequential number of the attachment and *filename.ext* is the original file name with extension
- *description* is an optional attachment description
- *data* stores an attachment content encoded according to data URI (http://en.wikipedia.org/wiki/Data_URI_scheme) scheme, if file is encrypted then attachments must be embedded
- *file* file to an external attachment file (only for decrypted files)

Traceability

Traceability section stores an array with links configuration between documents.

```
{
  linkTypeId: {
    name:       "Reference",            // optional
    source:     "References",           // optional
    target:     "Referenced by"         // optional
    relations:  [                       // optional
      { source: "UR", target: "UR" },
      ...
    ]
  }
}
```

Where:

- *linkTypeId* is link type identifier, for instance *reference*
- *name* is link type name
- *source* describes the role of the link source
- *target* describes the role of the link target
- *relations* is an optional array of possible relations between documents. If *relations* is not specified then all combinations of source and target documents in the project are allowed. Otherwise all pairs of linked documents with given relation needs to be listed explicitly.

Each link is stored at source object of the link as property named by *linkTypeId*. See example of a link from *38* to *25* in Section Data.