ALPHAREN CORE-Integrator (ARINT) System

System Data and Objects

product version: `0.1`

Table of contents:

Introductory

Naming rules

- Objects `code-name` is not case sensitive and is a good rule to use always uppercase. The system will convert the data objects `code-name` to uppercase, even from the save step.
- New "user or client or implementation" level business objects `code-name` should start with character `Z`. Any other ARINT objects do not use `Z` as first character in code-names, so this can make an *acceptable separation of those objects that are subject of change by system updates* and is a *guarantee that objects starting with* `Z` *in code-name will not be altered* by any system update.

Object categories and models

Any data object must be in one of these categories:

- **entity** - This is a relational data entity, more exactly a *table* to use the specific relational modeling concept. Such kind of objects are stored in relational system database.
- **json** - This is an object can contain any data model and type that respect `JSON` standard. This kind of object can cover almost all kind `NoSQL document type` variants. Such kind of object can be stored in any system database (relational or kv store) at your option. Default store place is kv system database.
- **crypto_key** - This kind of object contains keys used for cryptographic purposes. This kind of object is represented as a limited `json` type (strict one level, no arrays, only string type keys). A `crypto_key` object usually has one or two entries corresponding to one (in symmetrical cryptography) or two (in asymmetrical cryptography) keys, both as strings of `UTF-8` characters (bytes).
- **file** - This kind of object is designed to keep files or directories references. This references are stored can be stored in any system database (relational or kv store) and must respect all *Linux* conventions regarding file names (directories are files too represented as convention with a `/` character at end if is not obvious from context). So as a minimum strings of `UTF-8` characters (bytes) no more than 256 characters (bytes)

As definition models the `OO` definition is preferred and for relational objects the *SQL Alchemy* is preferred option. For `kv` data the Python normal `dictionaries` can be used.

Namespaces catalog

Here are listed the main *area of objects* defined in *System database*. These are the equivalent of a `namespace` (or `schema` in relational databases terminology) and they will become in a way[1] a `prefix` for all entities / objects contained in.

- **URRM** user roles and rights management
- **ODEF** business objects definition
- **MADA** global master data (cross any systems, applications, nodes, ie global for an **ARINT instance**)
- -#NOTE future reserved ...

Namespace URRM

URRM catalog

- -#NOTE future reserved ...

URRM_new_object_x

- -#NOTE future reserved ...

Namespace ODEF

ODEF catalog

- -#NOTE future reserved ...

ODEF_new_object_x

- -#NOTE future reserved ...

Namespace MADA

MADA catalog

- `OBTYP` object types and models. This contains values described of "Object types and models" section
- -#NOTE future reserved ...

MADA_new_object_x

- -#NOTE future reserved ...

Notes and remarks

---

1. (`PROPOSAL @ 230823 by piu`): The way that concrete entities / objects are prefixed depends of database, ORM instrument used, etc. To keep as agnostic as possible this prefix will be used as *object name prefix* followed by `_` character. **All `code-name`s for a prefix will be kept in 4 characters**. ↩

Last update: August 23, 2023