Kernel

Kernel provides the lowest level of functionality on which ORMS and the rest of the Prefiniti system relies.

UUID

- **Provides support for universally unique identifiers**
- **Defined in src/krnl/KRNLUUID.m**

RNSI

- **Provides support for I/O on roll-and-scroll terminal interfaces**
- **Defined in src/krnl/KRNLRNSI.m**

ORMS

ORMS records provide fully-versioned, ACID-compliant data storage for all Prefiniti applications and modules. Each ORMS record is keyed to an instance-specific unique identifier, and can contain an unlimited number of key/value pairs representing individual data items.

Record

- **Provides support for versioned key/value records**
- **Defined in src/orms/ORMSREC.m**

Every ORMS record may define a **parent record**, which can either be the ID of another arbitrary record, or the ID of the root record for the instance, which must be defined by the instance administrator, and may be obtained by calling **$$ROOTNODE^ORMSREC()**.

Every ORMS record may also associate itself with a **class**, which is a higher level of abstraction than a record, defining a set of key/value pairs for a particular grouping of data objects, along with metadata information for each key/value pair (such as data type, field description, maximum length, and whether or not the pair is required for a particular instance of a class to be considered complete), allowing the class compiler to produce appropriate device- and interface-independent forms for initial data entry, revision, and deletion.

Every key/value pair within an ORMS record is **versioned**: that is, when any pair's value is revised, the prior revision is left intact, and a new key/value pair with an incremented revision number is stored. Thus, each individual revision of a pair can be considered immutable, in that its data is never modified. Growth of any pair's revision history may be controlled by calling **CULL^ORMSREC(UID,COUNT)**, which deletes old revisions of the pair's data, while preserving at most **COUNT** revisions.

Every revision of every key/value pair also stores the date and time at which it was created (in MUMPS $HOROLOG format), as well as the UNIX process ID of the Prefiniti process which created it.

Every ORMS record may be locked by calling **SETLOCK^ORMSREC**, which will prevent all processes (with the exception of the process owning the lock) from revising or deleting the record.

When a record is created, revised, or deleted, ORMS will asynchronously execute all registered hooks for that record's class type. Hooks run in a process separate and distinct from the process performing the action on the record. Thus, hooks may only revise a record's data if the record has not been locked.

Class

- **Defines a set of ORMS records for a particular grouping of data objects**
- **Defined in src/orms/ORMSCLAS.m**

Classes define a set of key/value pairs for a particular grouping of data objects, along with metadata information for each key/value pair (such as data type, field name, maximum length, and whether or not the pair is required for a particular instance of a class to be considered complete), allowing the class compiler to produce appropriate device- and interface-independent forms for initial data entry, revision, and deletion.

Every ORMS class definition is itself stored as an ORMS record of class **CLASDEFN**, which allows every class to take full advantage of ORMS versioning facilities and hooks. Thus, the ability to revert to a prior version of a class when new modifications cause unexpected behavior is built in at the lowest level.

Hook

- **Allows developers to define per-class routines to be called when records in a class are created, revised, or deleted**
- **Defined in src/orms/ORMSHOOK.m**

Any Prefiniti application or module may register any number of **hooks** against a particular ORMS class, which instructs ORMS to execute a programmer-defined MUMPS subroutine when any key/value pairs within the class are created or revised, or when the entire record is deleted. The conditions under which the hook will be executed (that is, CREATE, REVISE, or DELETE), are defined in the hook's **event mask**.

When a key/value pair is created or revised, or deleted, ORMS will asynchronously execute all registered hooks for the containing record's class type matching the hook's event mask. Hooks run in a process separate and distinct from the process performing the action on the record. Thus, hooks may only revise a record's data if the record has not been locked.

Every hook registered will be notified of the hooked record's ID, the key of the created, revised, or deleted record, its revision number, and the new data (if applicable).

Hooks are the lowest-level ORMS method facilitating the Prefiniti event-driven programming model.

Example Application for Hooks

In this example, the ORMS hook facility is used to notify an employer by e-mail of employees' change of address.

We will assume the following:

- The application has defined a class, **EMPLOYEE_ADDRESS**, which contains key/value pairs for street address, city, state, and ZIP code.
- The application has registered a **REVISE** hook against **EMPLOYEE_ADDRESS**.

In this example, the hook would be called when any key/value pair belonging to class **EMPLOYEE_ADDRESS** is revised. The hook would receive the new address information, and be able to then e-mail it to the employer.