|  |  |  |
| --- | --- | --- |
| **SIGNUP** | | |
| Object representing whole signup | Actium::Signup | Modification of Actium::O::Folders::Signup, to just return folder objects when folders are wanted |
| **SCHEDULE – of a line** | | |
| Object representing a schedule | Actium::Sked | Mostly the same as Actium::O::Sked, with some things moved to subsidiary objects |
| Object representing a collection of schedules | Actium::Sked::Collection | Since signups have several collections of schedules (at least received, final, exceptions), this can’t be just a role. |
| Object representing the stop of a schedule | Actium::Sked::Stop | This contains the stop ID and the place ID (if any) associated with the stop in the parent Actium::Sked object. Takes the place of stoptimes, stopplaces, etc. in Actium::O::Sked. The stop ID points to the Actium::Stop object |
| Role representing a collection of schedule stops | Actium::Sked::Stop::Collection | Each schedule has exactly one set of stops, so this can be a role that is applied to Actium::Sked |
| Object representing each trip of a schedule | Actium::Sked::Trip | Mostly the same as Actium::O::Sked::Trip |
| Role representing a collection of trips | Actium::Sked::Trip::Collection | Each schedule has exactly one set of trips, so this can be a role that is applied to Actium::Sked |
| Object representing a time on a schedule | Actium::Sked::Trip::Time | This contains an Actium::Time object, but also information on whether it is interpolated by Actium, provided as part of the import, or whether it is actually doesn’t stop there at all for this trip. |
| Role representing a collection of times | Actium::Sked::Trip::Time:: Collection | Each trip has exactly one set of times, so this can be a role that is applied to Actium::Sked::Trip |
| **IMPORT – Modules representing various formats imported into Actium** | | |
| Object representing a Xhea import | Actium::Import::Xhea |  |
| Objects for each Xhea table | Actium::Import::Xhea::\* |  |
| Object representing a GTFS import | Actium::Import::GTFS |  |
| Objects for each GTFS file | Actium::Import::GTFS::\* |  |
| Objects/routines shared among import formats | Actium::Import::\* |  |
| **STOP** | | |
| Object representing a bus stop | Actium::Stop | Contains the information related to a bus stop: what lines stop there and when. |
| Object reprsenting schedules at a bus stop | Actium::Stop::Sked | Contains the schedule information just about that stop: the times here, what the destination is of each route that stops here, etc. Similar to what’s in kpoints, but also flag information. |
| **SITES AND SIGNS** | | |
| Object representing a point in space that should have information | Actium::Site | Distinct from a stop in that it includes places that are not stops (e.g., TID locations). Distinct from a sign in that multiple physical signs can be placed there (e.g., when two or more pole schedules or TIDs are needed). |
| Object representing a physical sign (with a paper insert) | Actium::Sign |  |
| **COMPARISON OBJECTS/CLASSES** | | |
| Comparison between two schedules | Actium::Compare::Skeds |  |
| Comparison between sets of stops | Actium::Compare::Stops |  |
| **DATABASE** | | |
| Object representing Actium database | Actium::DB | This represents the content of the Actium database rather than the underlying database mechanism. Anything that wouldn’t change if the database were suddenly moved from one database software to another. |
| Objects representing Actium database tables | Actium::DB::Stops, Actium::DB::PubTimetables, etc. |  |
| Roles for common behavior among database table classes (such as caches) | Actium::DB::Table if just one, otherwise named for behavior |  |
| **APPLICATION ENVIRONMENT** | | |
| Environment object | Actium::Env | This is what is returned by Actium::env() |
| Command line interface class | Actium::CLI | The idea is that this would create the Actium::Env object |
| Command line options | Actium::CLI::Option | What is now Actium::O::Cmd::Option |
| Terminal input/output | Actium::CLI::Crier | If it isn’t hived off into its own distribution |
| Commands | Actium::Cmd | Actual command modules go into Actium::Cmd, as opposed to things that work on the environment such as Actium::CLI::Crier |
| **STORAGE** | | |
| Folder object (relates to Path::Class::Dir) | Actium::Storage::Folder |  |
| File object (relates to Path::Class::File) | Actium::Storage::File |  |
| FileMaker database class | Actium::Storage::FileMaker | Low-level database access |
| **MORE PRIMITIVE DATA TYPES** | | |
| Time object | Actium::Time |  |
| Date object (to compare dates) | Actium::Date |  |
| Days of service object | Actium::Days |  |
| Direction object | Actium::Dir |  |
| Text object and classes | Actium::Text | Used for multilingual output, InDesign or HTML tags, etc |
| Set / combinatorics | Actium::Set |  |
| Type library | Actium::Types | Probably should move to Type::Tiny |
| **Misc** | | |
| Map repository | Actium::MapRepository | We need to consider the larger issue of digital asset management. |