



ESO/L Calçada

Release Candidate Update

Tom Coppeto
April 28, 2013

Timeline

OSID V2 Release
(16 packages)

2004

2006

OSID V3 Launch

- initial MIT funding ~0.5FTE
- solve V2 pain points
 - authentication
 - server-side operation
 - out-of-band agreements
 - broader design input

OSID V3 Draft 3
(24 packages)

- first development tools
- first V3 implementation
- end of MIT funding

2008

Timeline

MIT Learning Architecture

2010

OSID V3 Draft 6
(42 packages)

- more extensive developer kit
- OSID Runtime and configuration service

2011

MC3 Project Launch

- gives Draft 6 a go

Timeline

Current

- bug fix only
- vetting and testing
- strives for interface stability

2012

OSID 3K RC (162 packages)

- major cleanup
- “big league” patterns
- divorce spec from contrib software
- strives for interface stability
- osid.org launch

2013

Purpose

- * Remains to promote interoperability through the use of software interfaces
 - * interoperability = ability to swap
- * New themes:
 - * ability for OSIDs to be used as primary design and development tool as opposed to an add-on
 - * ability for OSIDs to be used to develop enterprise class systems once piece at a time

Focus

- * Avoid moving trains
 - * projects cannot use works in progress
- * Be good at something
 - * integration problems in Java
- * It is what it is
 - * it's useful to you or it isn't

Approach

- * Walk before run
 - * small bite-size problems
 - * hundreds of interoperability cases is better than a couple of flagship projects
- * Stay off the pulpit
 - * sell solutions, products, and services, not architecture, models, or interfaces

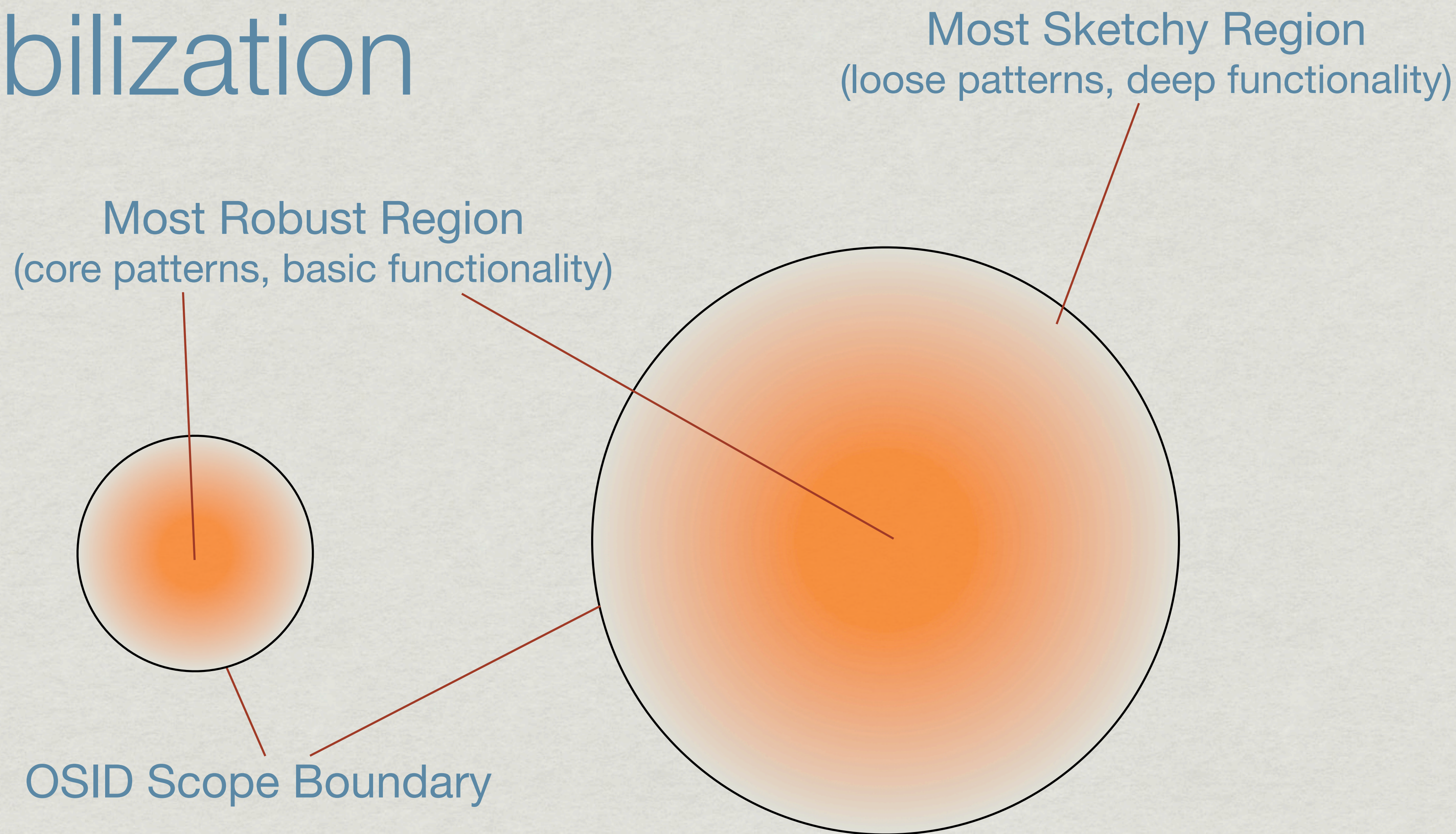
Stabilization

- * Extensibility mechanisms serve many use cases not addressed in core spec
- * Auxiliary services may act as building blocks to supplement an OSID.
- * RC pushes the envelope in modeling more services and patterns to stabilize the spec
- * The ultimate in prefactoring!

Building Confidence

- * Instability results from lack of confidence and lack of diverse use cases
- * It isn't so much about analyzing the weeds
- * It's more about knowing where a problem is to be solved
 - * OsidRecords fill in details needed by an SOR
 - * The “places” (entities) standardize the pathways
 - * Confidence increases with the number entity relations across various service domains

Stabilization



Increasing model and pattern interdependencies increases stability.

Deja Vu

- * If it applies here, it applies everywhere else
 - * repeating service patterns now get applied whether or not it is identified as a need at the time
 - * avoids nickel & dime contract changes
 - * identifies concepts to change thinking about the business application
- * Building models on the backs of others
 - * natural but unforced reusability is a good sign
 - * abstraction is an art, not a mantra
 - * creates many a-ha moments when things “click”

Circling Pluto

- * Far out use cases to test the flexibility
 - * designing to immediate scope results in ever changing contracts
 - * OsidPrimitives are an important (and somewhat humorous) flex point that does more to help reusability of a service model than anything else
- * Reducing the gaps
 - * OSIDs expanded to reduce the conceptual “jumps”
 - * should be able to pick any point in a service model and be able to circle back to it, then do it in the opposite direction. That’s the “snap.”

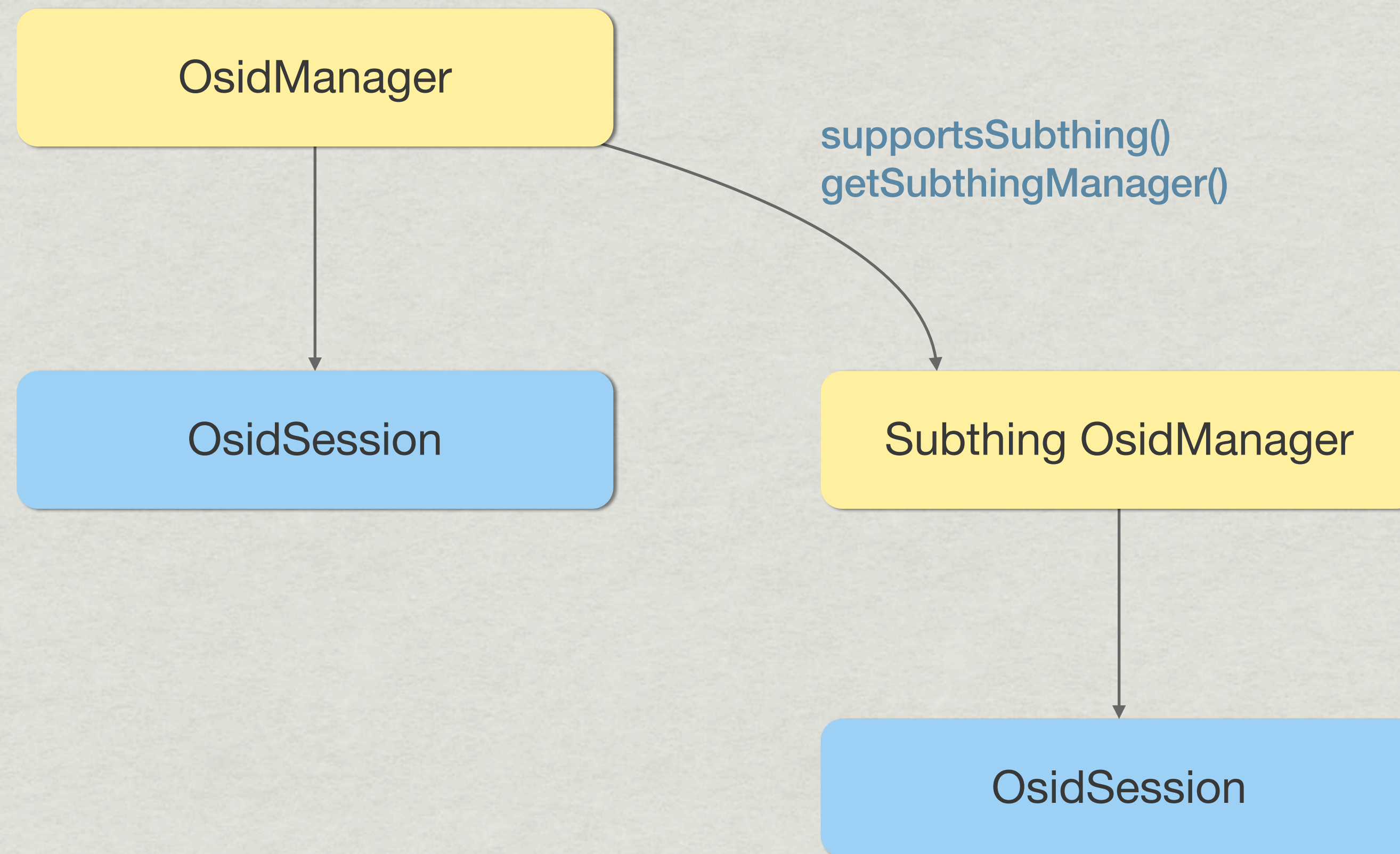
RC Headlines

- * packaging
- * “bite the bullet” pattern changes
 - * redesign of interface semantics
 - * richer Locale
- * searching & magic catalogs
- * batch services
- * rules rules

Packaging

- * osid.org only for the specification
- * OSID Packages can be nested
 - * Cluster and hide advanced functions from top-level view
 - * The nested OSID is only visible through the OsidManager of the parent
 - * new clusters of functionality can be added to an OSID with minimal disruption

Sub Packages



Locale

- * Locale clumps together localization Types
 - * Allows for constrained sets
- * OsidSessions and OsidForms use Locales instead of Types

osid.locale.Locale

- + Language Type
- + Script Type
- + Calendar Type
- + Time Type
- + Currency Type
- + Unit System Type
- + Numeric Format Type
- + Calendar Format Type
- + Time Format Type
- + Currency Format Type
- + Coordinate Format Type

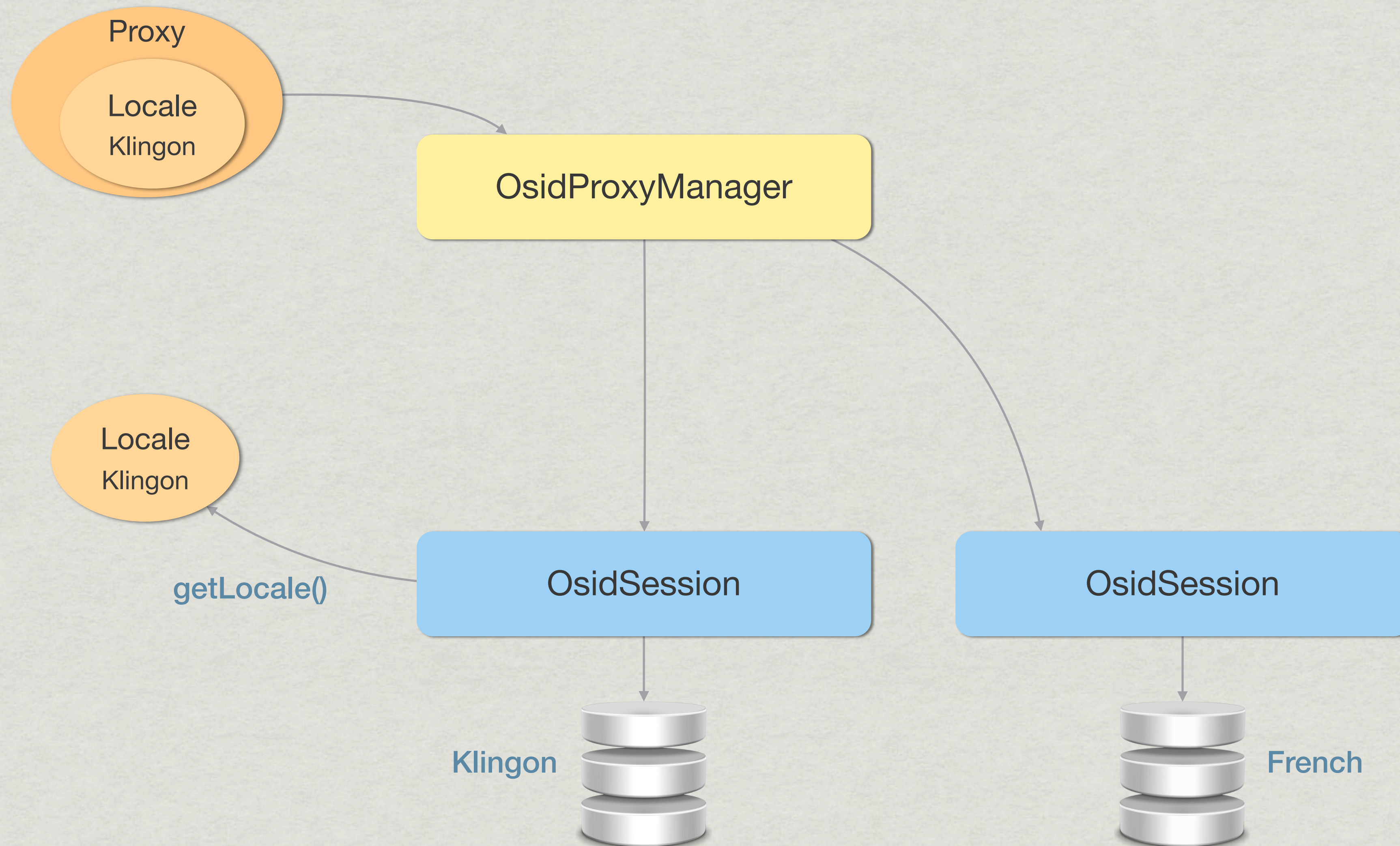
DisplayText

- * Strings no longer used for display
- * Facilitates service adapters

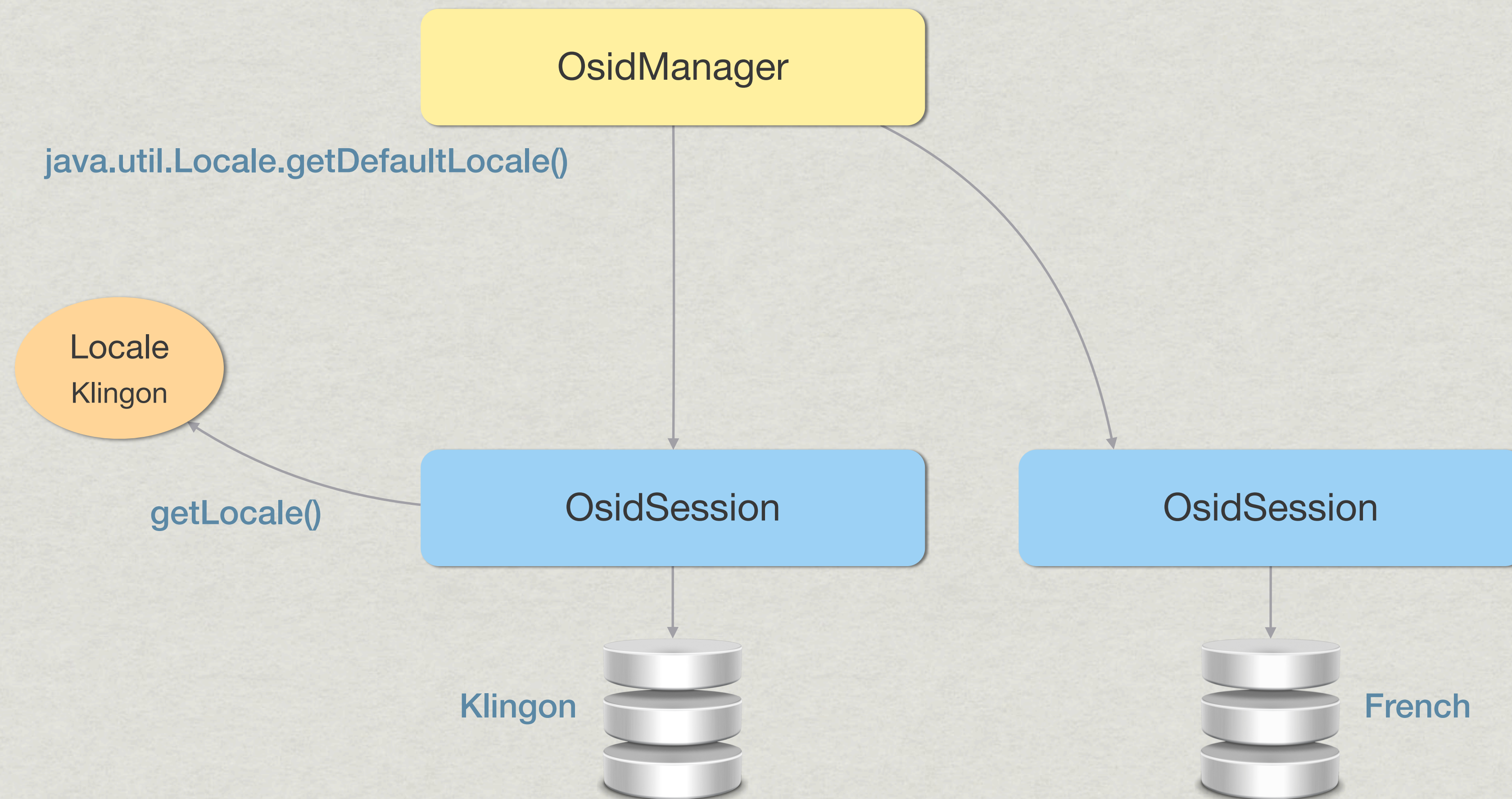
[osid.locale.DisplayText](#)

+ Language Type
+ Script Type
+ Format Type
+ Text

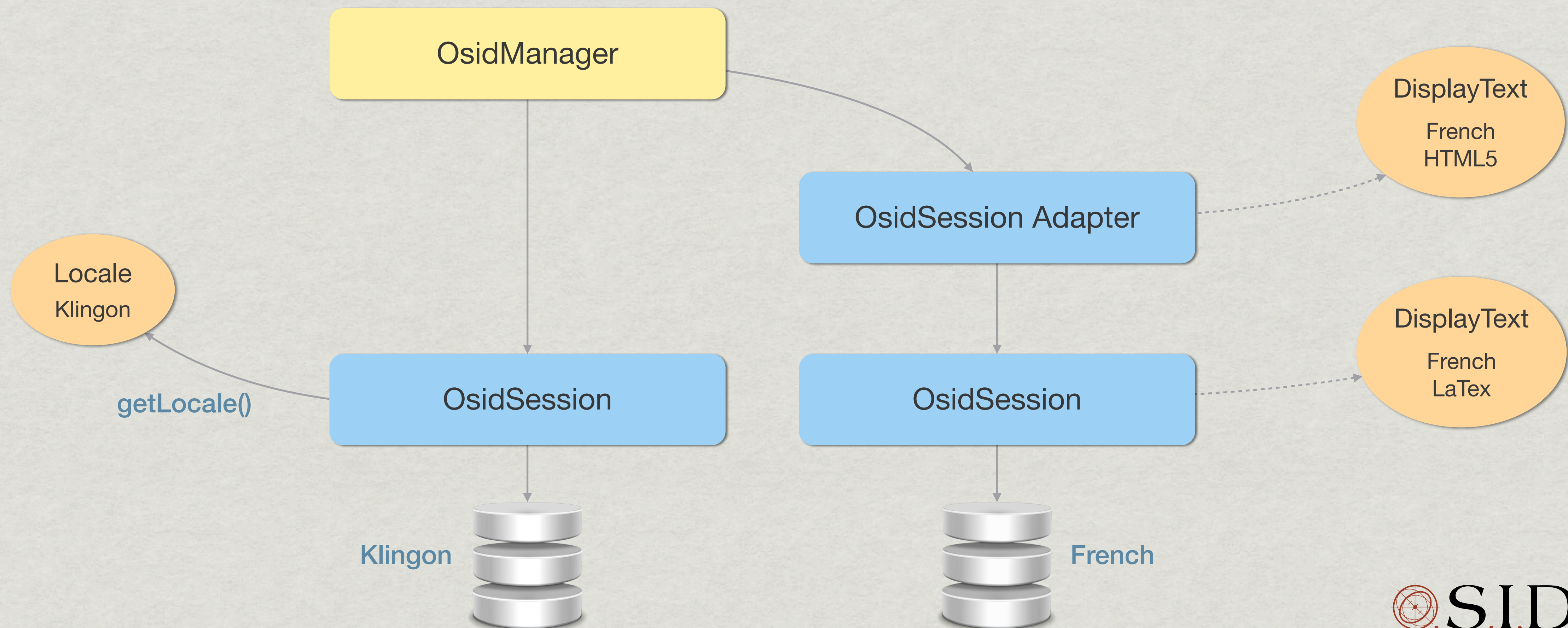
Localization Pattern



Localization Pattern



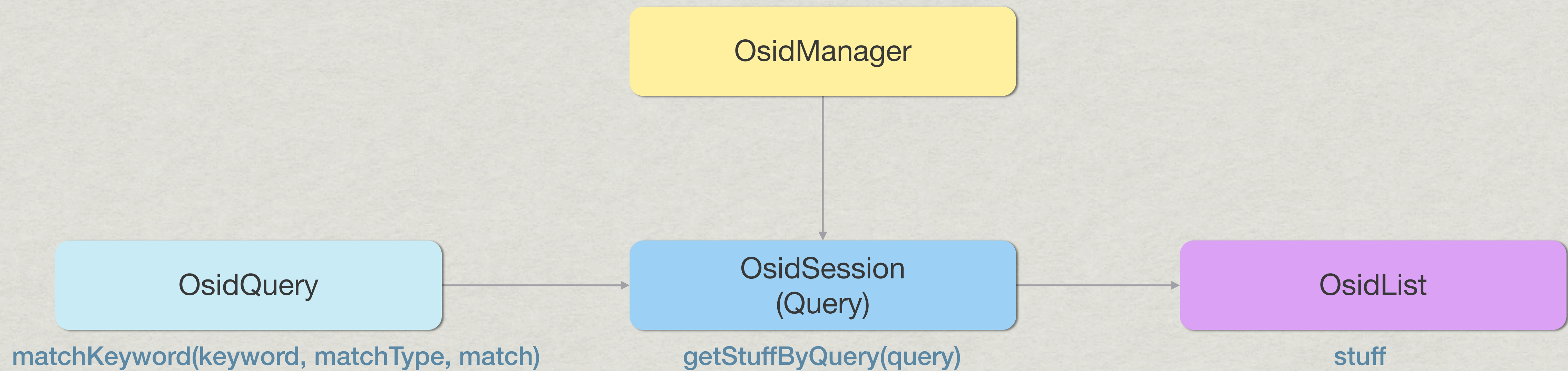
Localization Pattern



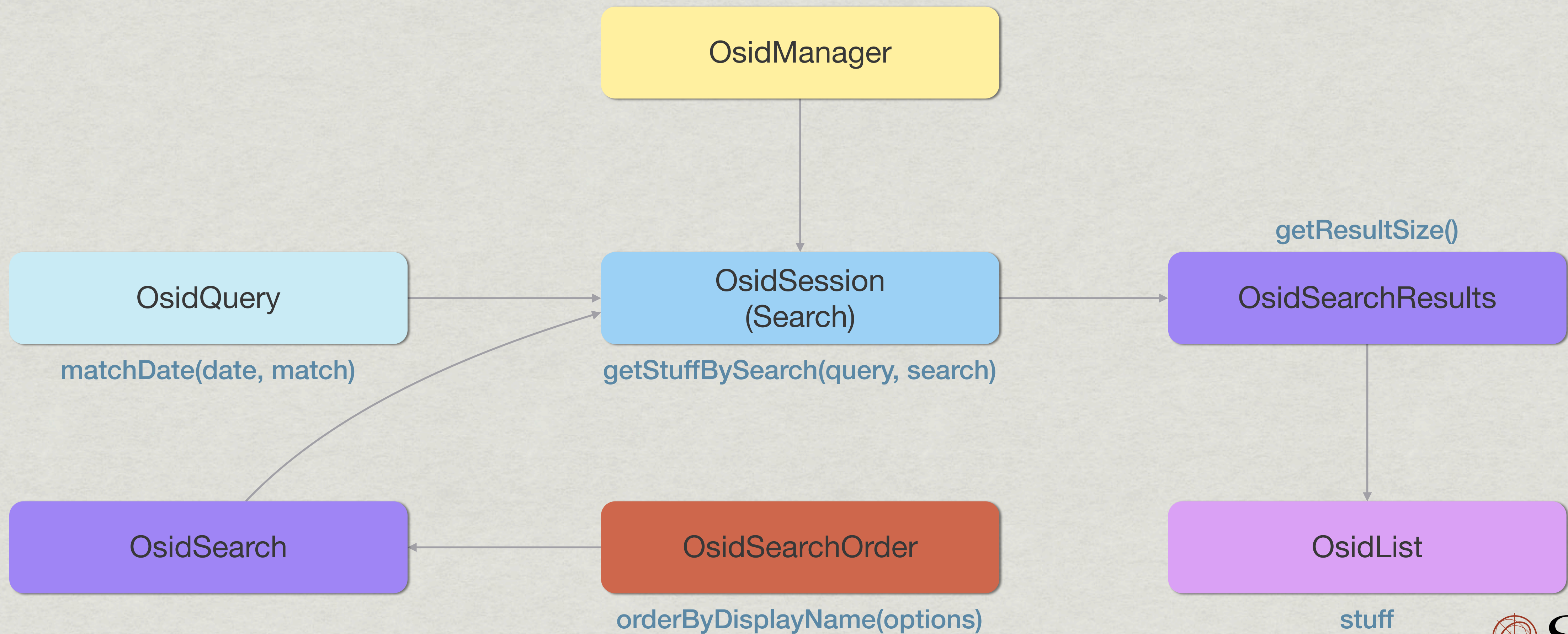
Searching

- * Searches hard to implement
 - * Queries focus on simple matching
 - * Searches do that and have a dialog with a search engine and make an attempt at ordering
- * Split into separate OsidSessions
- * Search OsidSessions extend query OsidSessions
- * abstract pattern in OSID Search service

The Short of It



The Long of It



OsidQueries

- * OsidQueries support cross-OSID queries
- * Out-of-band orchestration with auxiliary OSIDs
 - * OsidObject queries define matches for States, Relationships, Credits, Statistics, Journal Entries, and Comments

OsidSearchResults

- * State collision

- * this interface wraps the matched items plus any results from the search engine
- * what happens when the stream of matched items is retrieved twice for the same transaction?
- * the “fix” was to redefine it as a once-only retrieval and add an ILLEGAL_STATE error

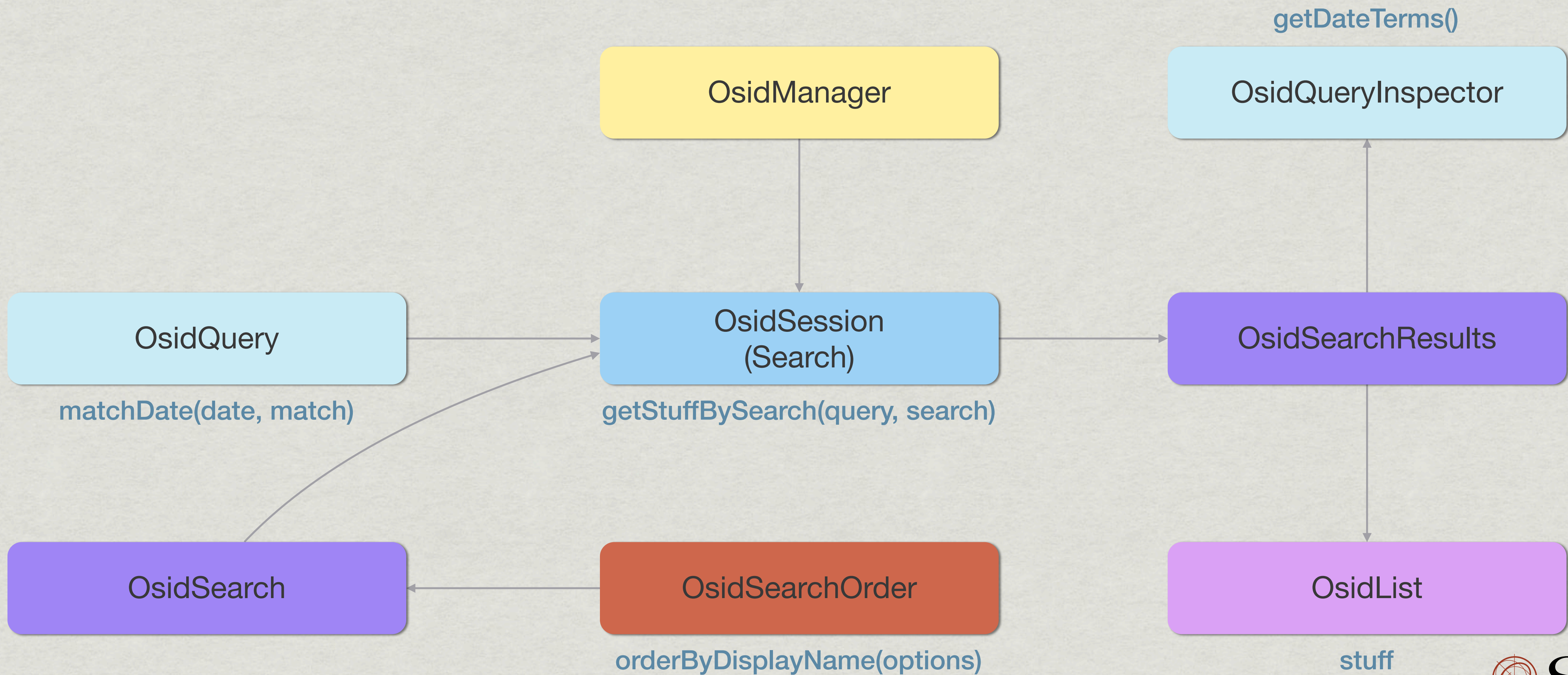
Black Box Queries

- * OsidQueries capture the desire of an OSID Consumer
- * there is no guarantee that any of it is honored by an OSID Provider
- * so, what really happened?

OsidQueryInspectors

- * OsidQueryInspectors are available in OsidSearchResults
- * OsidQueryInspectors provide information about the actual query executed on a term by term basis

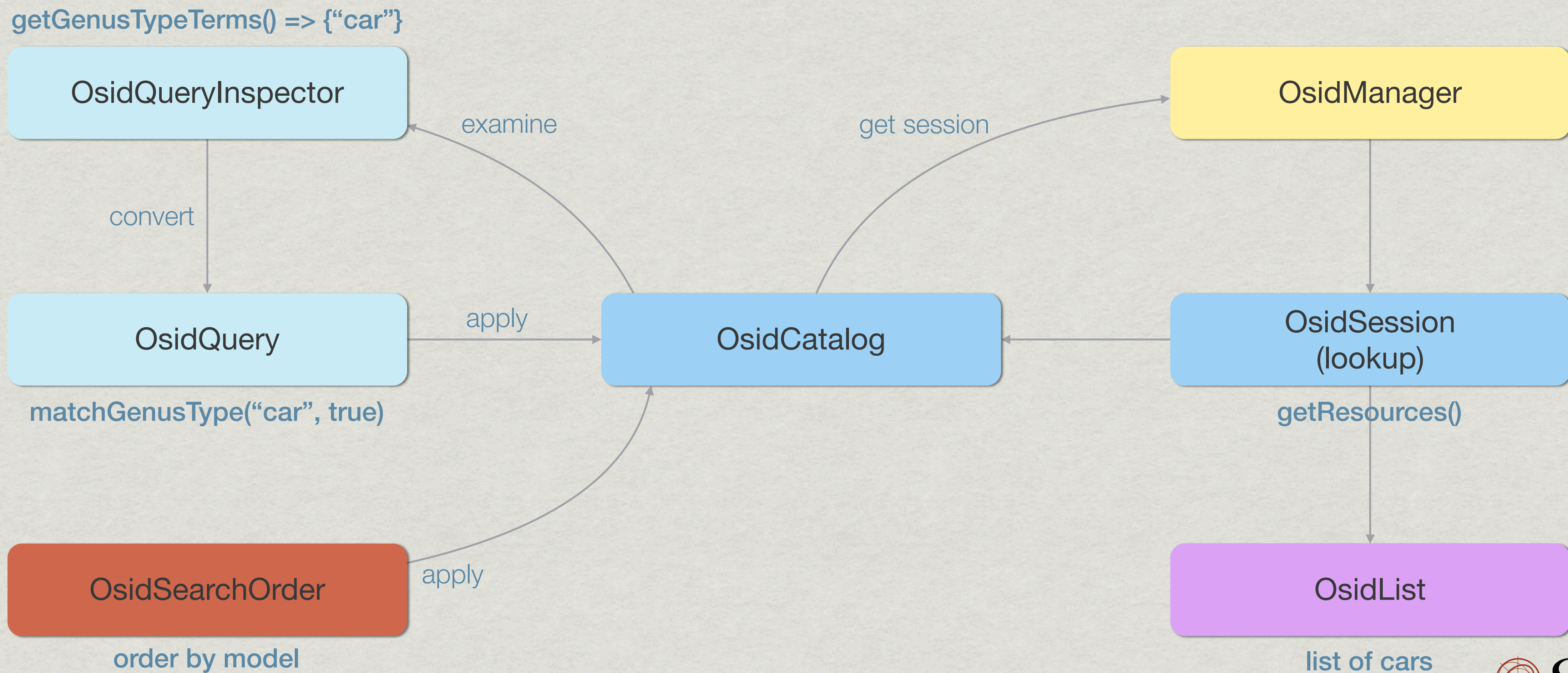
The Longer of It



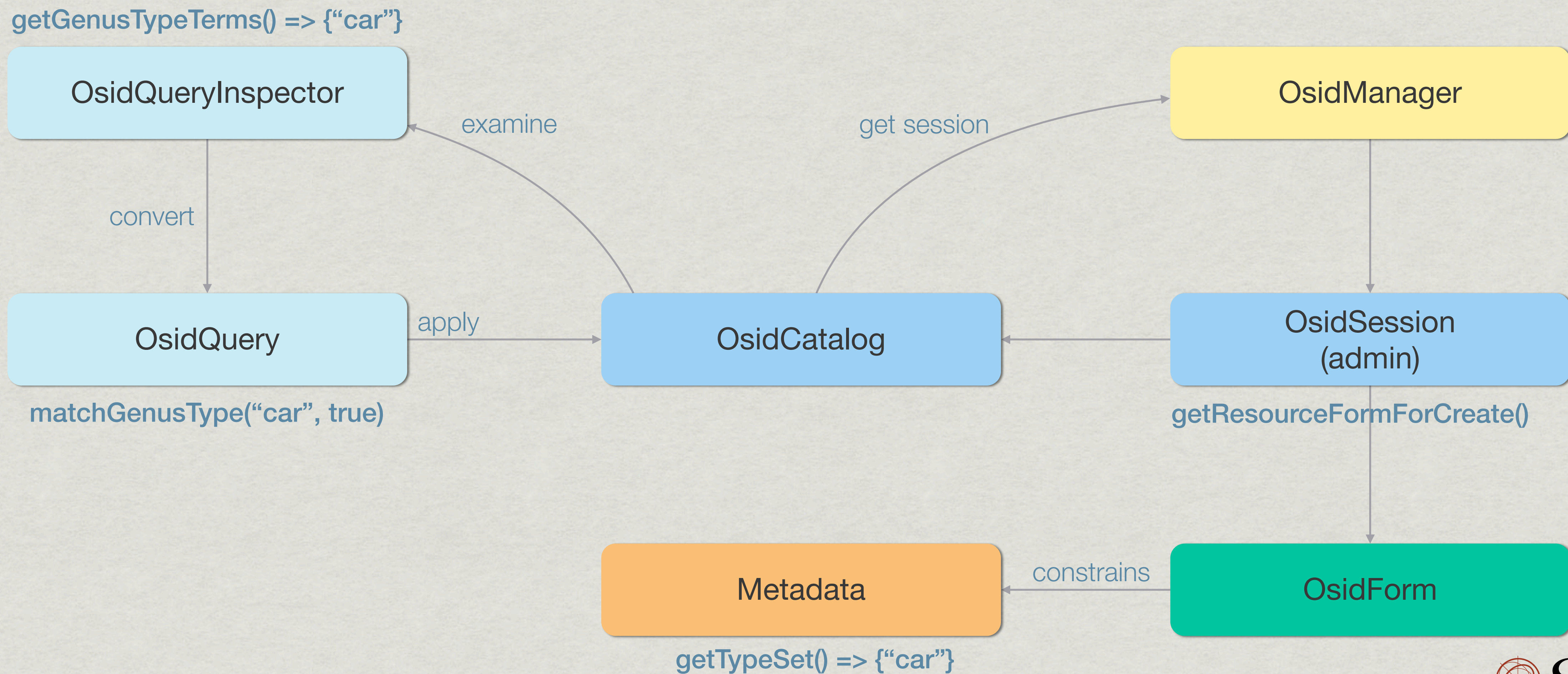
Strange Days

- * So, what happens when we apply an OsidQuery to an OsidCatalog?

Magic Catalogs



Rabbit out of a Catalog



OsidCatalogs

- * Queries can be inspected and applied to OsidCatalogs in smart catalog OsidSessions
- * All OsidCatalogs have an implicit OsidQuery of “match anything in any order”

Two Ways To Do The Same Thing?

- * OsidQueries can be used to constrain what is visible (filter) or what is created (constraint)
- * Wait, that's what Types were used for!
- * The smart catalog sessions expose the authoring of these *rules*

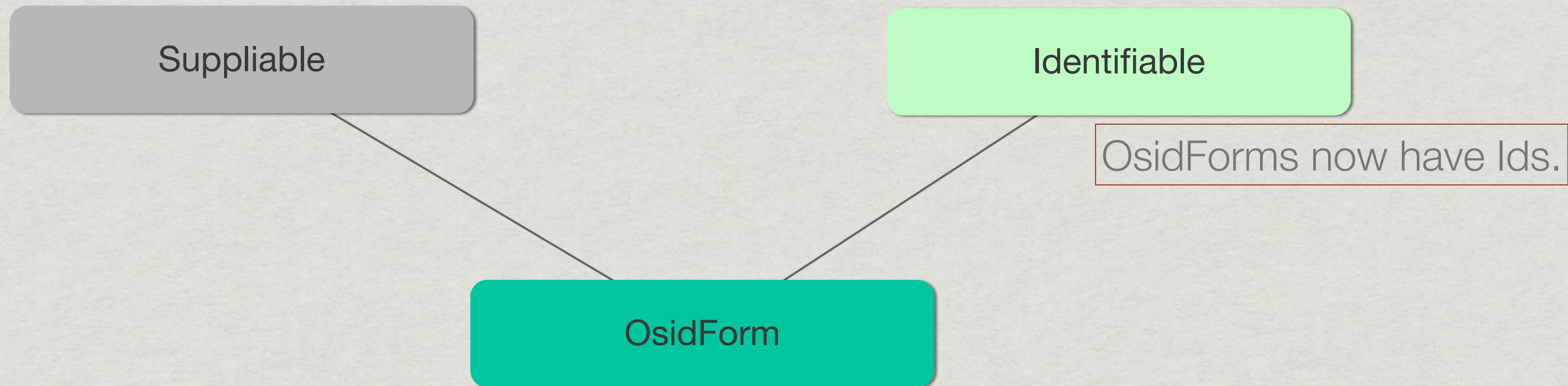
An Identifier By Any Other Name

- * If a genus Type is used for categorization or data constraint, it is essentially shorthand for this more elaborate mechanism
- * Both genus Types and OsidCatalogs refer to a class of OsidObjects
- * A genus Type is a means of forging agreements around OsidCatalog Ids
- * Dizzy?

Admin OsidSessions

- * Removed pre-auths for single objects
- * All create parameters pushed into OsidForm retrieval
 - * acquiring the OsidForm is a complete service operation that is responsible routing and determining metadata for validation for creates and updates
 - * added OPERATION_FAILED error

OsidForms



```
CourseOfferingForm form;  
form = getCourseOfferingFormForCreate(Id courseId, Id termId, Type[] recordTypes);  
CourseOffering offering = createCourseOffering(form);
```


Batch Services

- * Create, update, delete, and alias OsidObjects in bulk
- * Address efficiency worries with data feeds
- * In sub-packages across all OSIDs
 - * Batch Admin OsidSessions extend Admin OsidSessions in the parent package
 - * Batch OsidForms extend OsidForms in parent package

OsidForm Acquisition

- * OsidForms are retrieved in bulk
 - * one set of record Types per retrieval
 - * delivered via an OsidList
- * One OsidForm per create or update transaction (why they are now Identifiable)
- * For updates, an IdList of OsidObjects to be updated is needed
- * For creates, that gets a bit more complicated

Batch Creates

- * If no create parameters, then the number of OsidForms is specified
- * If one create parameter (dependent object), then an IdList is supplied
- * If two create parameters (relationship), then we need another interface

Batch Peers

- * The peer interfaces simply capture the Id pairings for bulk OsidForm retrieval
- * The peer interfaces are provided by the OSID Consumer and consumed by the OSID Provider
- * Peer interfaces are also supplied using an OsidList

Batch Create Examples

```
MyMutableResourceBatchFormList outputForms = new MyMutableResourceBatchFormList();

try (ResourceBatchFormList inputForms = getResourceBatchForms(99, desiredRecordTypes)) {
    int i = 0;
    while (inputForms.hasNext()) {
        ResourceBatchForm form = inputForms.getNextResourceBatchForm();
        // check metadata
        form.setDisplayName("resource #" + Integer.toString(i++));
        outputForms.addOutputForm(form);
    }
}

outputForms.doneAddingStuff();
CreateResponseList responses = createResources(outputForms);

// examine responses
```

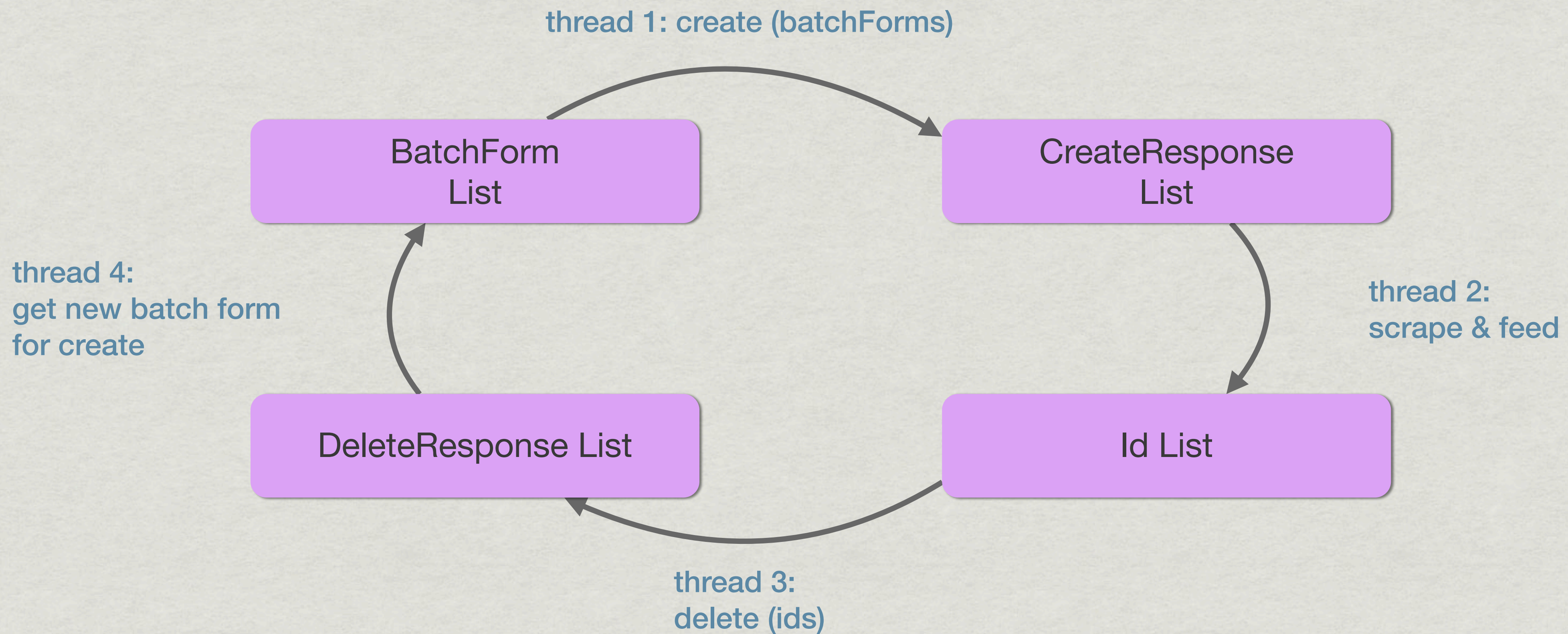

Batch Deletes

- * Both the batch create OsidForm retrieval and the batch delete operations follow the pattern seen in the lookup OsidSessions
 - * `getCourseOfferingsByCourse(courseId);`
 - * `getCourseOfferingBatchFormsForCreate(courseIdList);`
 - * `deleteCourseOfferingsByCourse(courseId);`
 - * there is also `deleteCourseOfferings()` and `deleteCourseOfferingsByIds(courseOfferingIdList)`

Fine Grained Deletes

- * Could do a query, and feed the resulting Ids into `deleteObjectsByIds(ids)`
- * Could also create a smart `OsidCatalog`, check the results, then delete everything
- * The `OsidCatalog` encapsulates filtering and validation rules exposed through the `OsidQuery`, so, either way

Closing the Loop



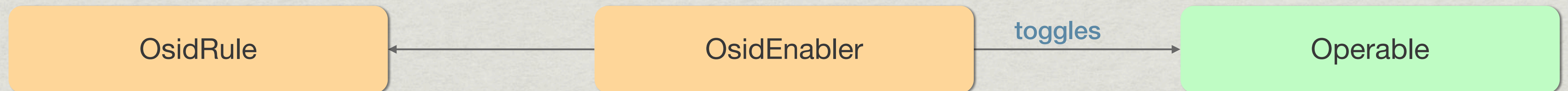
Rules

- * Rules breathe life into a cruddy system
- * OsidRules are OsidObjects with external “rule” attachments
 - * the Rule Id is a reference to a Rule in OSID Rule package
 - * OSID Rule is an abstract interface for a rules engine
- * OsidRules are Operables

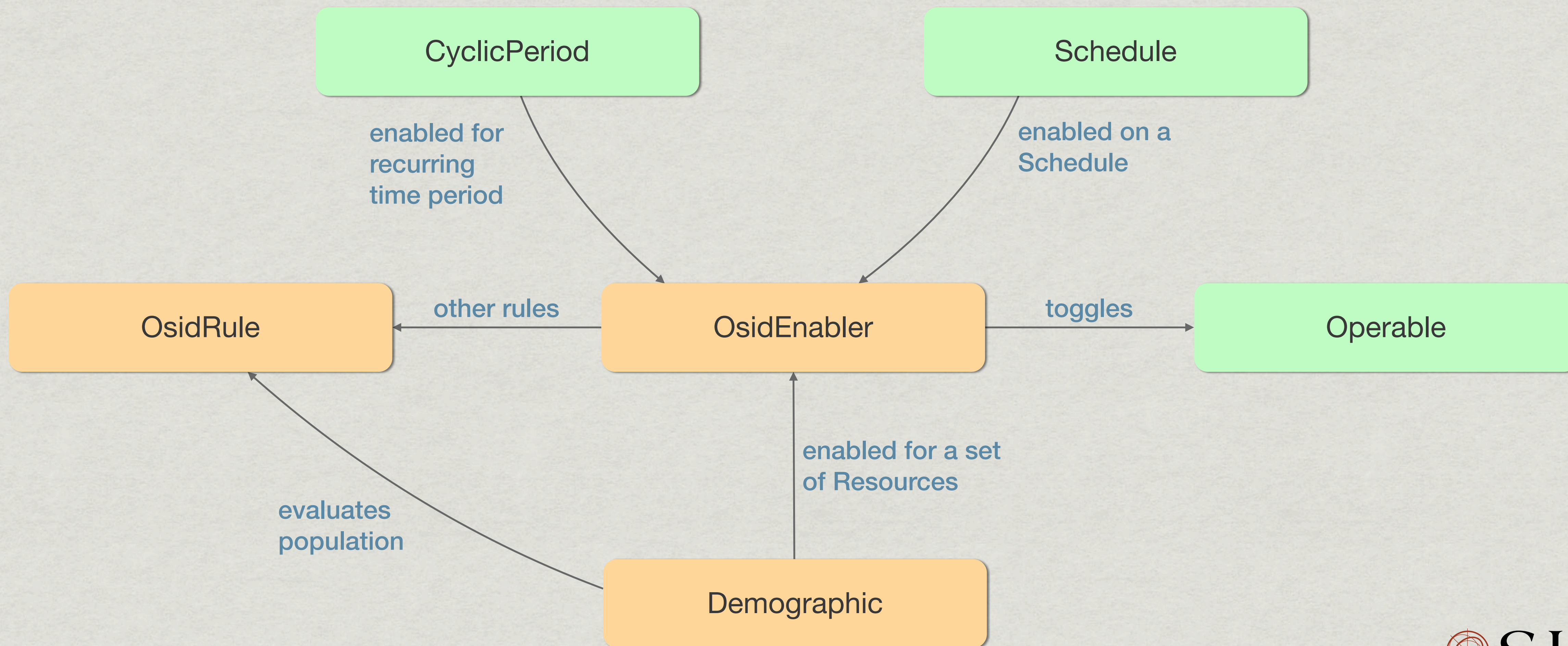
Toggling OsidRules

- * Operables can be manually operated: turned on and off
- * OsidRules can be automatically operated through OsidEnablers
 - * OsidEnablers are also OsidRules that turn other Operables on and off
 - * OsidEnablers are managed in rules sub-packages

OsidEnablers



OsidEnablers Built-In Rules



OsidConstrainers

- * An OsidConstrainer is an OsidRule used where an evaluation occurs to be associated or “in” something
 - * a Queue is an example of something that might have an OsidConstrainer (“must be this tall to invade this queue”)
 - * OsidConstrainers have OsidEnablers
 - * Multiple OsidConstrainers may be defined but not operational as a result of the evaluation of its OsidEnablers

OsidProcessors

- * An OsidProcessor is an OsidRule that governs the processing of something
- * A Queue is also an example of something that might have OsidProcessors to manage getting off the queue
- * OsidProcessors also have OsidEnablers
- * Multiple OsidConstrainers may be defined but not operational as a result of the evaluation of its OsidEnablers

OsidGovernators

- * OsidGovernators are designated OsidObjects
 - * they are like OsidCatalogs where they represent sets of OsidObjects
 - * specifically, these are sets of things that have OsidProcessors
 - * as Operables, they govern the entire processing of an execution or workflow
 - * yes, they also have OsidEnablers

New OSID Packages

- * OSID Acknowledgement
- * OSID Bidding
- * OSID Billing
- * OSID Blogging
- * OSID Calendaring Cycle
- * OSID Checklist
- * OSID Commenting
- * OSID Communication
- * OSID Contact
- * OSID Course Chronicle
- * OSID Course Program
- * OSID Course Registration

New OSID Packages

- * OSID Course Requisite
- * OSID Course Planning & Syllabus
- * OSID Financials
- * OSID Forum
- * OSID Grading Transformation
- * OSID Hold
- * OSID Inventory
- * OSID Offering
- * OSID Ordering
- * OSID Personnel
- * OSID Recipe
- * OSID Recognition

New OSID Packages

- * OSID Resourcing
- * OSID Room
- * OSID Rules Check
- * OSID Search
- * OSID Tracking
- * OSID Voting
- * OSID Workflow

Related Info

- * osid.org
 - * Framework specification documents
 - * OSID Specification (HTML)
 - * Logical and OSID modeling diagrams
 - * OSID Java Binding (OSID Language Specification)
 - * javadoc
 - * jar
 - * Wiki & Issue Tracking: <https://www.assembla.com/spaces/osid-dev/>
 - * GIT: <git://git.assembla.com/osid.git>
- * Okapia OSID Java Development Kit:
 - * <https://www.assembla.com/spaces/osid-java-kit/>
 - * OSID Primitive implementations
 - * various implementation patterns
 - * OSID Runtime implementation
 - * various type identifiers

RC Implementations

- * Okapia OSID Runtime
 - * OSID Configuration
 - * OSID Rules
- * MIT Assessment (in progress)

Stuff Not There

- * Artifactory jar delivery
- * Complete distribution package
- * Comprehensive javadoc
- * Query and Jdbc Tools (in progress)
- * Admin tools
- * Orientation guide (pieces on older wikis)

Migrating from D6 to RC

* Branch!

- * all hell will break loose
- * classes that used to be under org.osid.impl have moved into one of 3 different projects, but not all of them made it yet
- * String -> DisplayText conversions
- * splitting of search session
- * changed create signature and removal of some preauths
- * jiggling of root interfaces
- * exception cleanup
- * abstract classes should buffer against new methods

