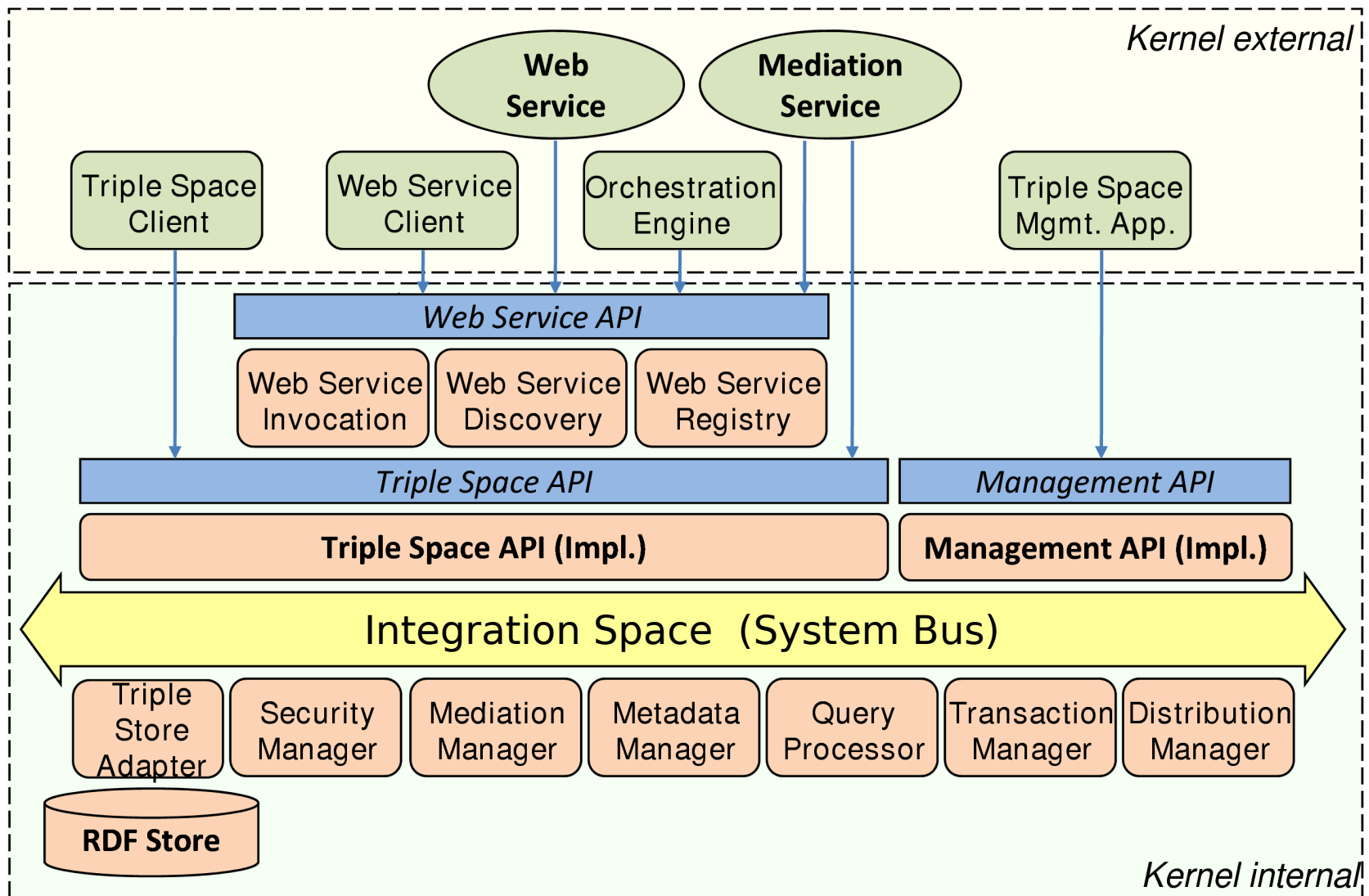
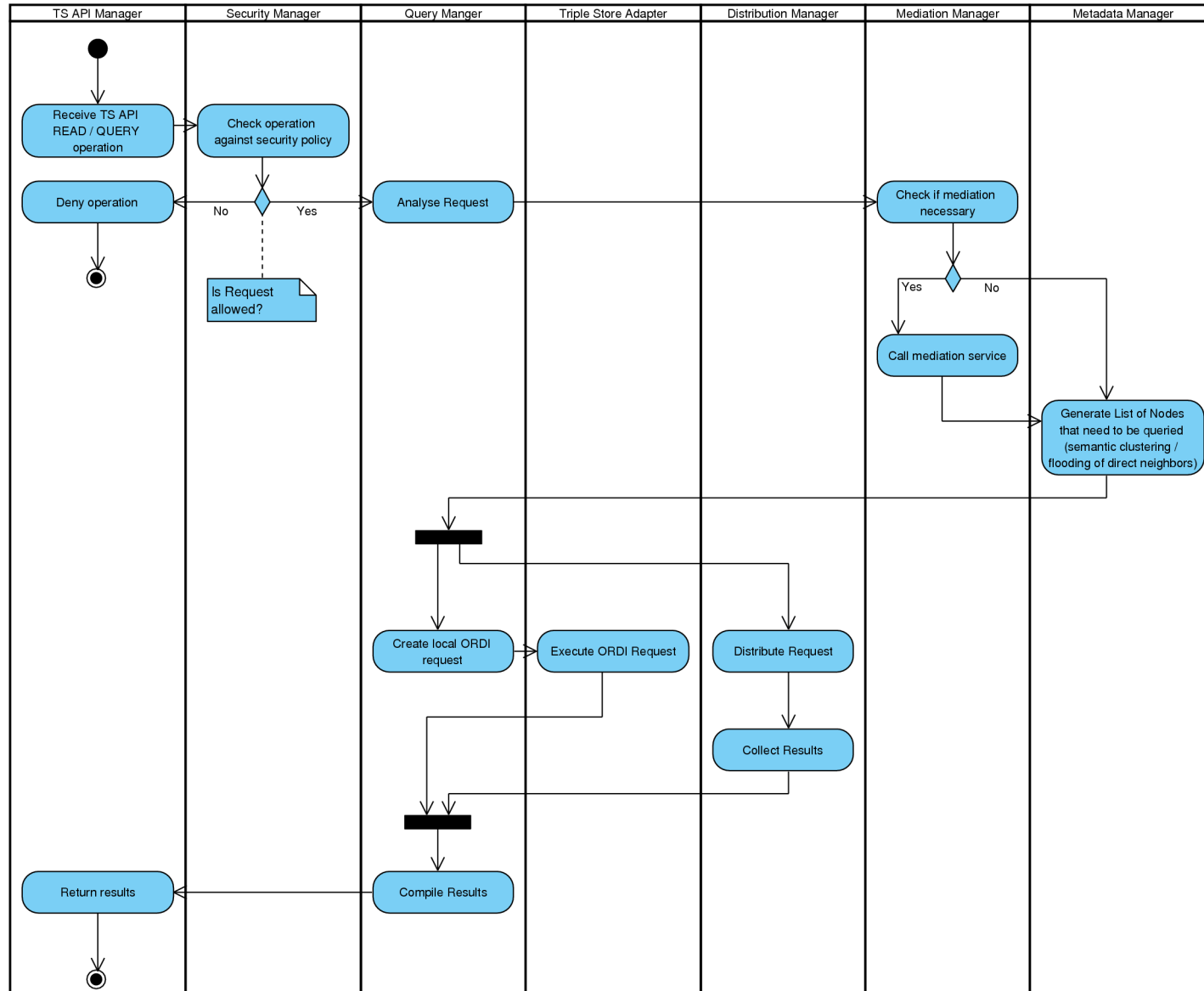


The 2nd TripCom prototype

Triple Space Architecture



Basic Component Interaction



- implements TS API operations
- client invokes a triplespace operation by submitting a request to the TS API implementation
- TS API operation is translated into an entry into the integration space that is communicated among kernel components.
- when the requested operation has been processed, the operation's result is returned to the client

- 1. Accepts requests from clients which conform to the given API
- 2. Returns responses to clients as defined by the given API
- 3. Handles grounding to concrete programming languages, transport protocols, data formats supported by client (abstracts from TS-internal implementation)
- 4. Handles exceptions caused by incorrect or inappropriate communication to the Triple Space

■ required Entries

■ DataResultExternal

- data result (triples) of a successful TS API rd or in operation optionally containing a security cookie

■ ErrorResultExternal

- error information in case of erroneous execution of a TS API Operation

■ provided Entries

■ WriteOpExternal

- representation of a TS API out operation, including security information

■ ReadOpExternal

- representation of a TS API rd or in operation, including security info

■ Provided

- Entry (1) writeop: to SecurityManager
 - Set<Statement> Data, URI Space, String ClientID, int OperationID, URI TransactionID, int Lease
- Entry (2) readop: to SecurityManager
 - TupleExpr Query, URI Space, String ClientID, int OperationID, URI TransactionID, int Lease
- Entry (3) transactionop: to SecurityManager
 - int OperationType, URI TransactionID

■ Required

- Entry (1) results: from SecurityManager
 - Set<Statement> Data, int OperationID
- Entry (2) errors: from SecurityManager
 - Set<int, String> Error, int OperationID

- representation of a TS API out operation
- including security information required by Security Manager to perform access control
- WriteOpExternal :=
 - tripleSet: Set<org.openrdf.model.Statement>
 - url: URL
 - time: Long
 - securityInfo: SecurityInfo :=
 - certificate (?)
 - Set<Assertion> (?)
 - cookie (?)
 - operationID: String
 - transactionID: URL

- representation of a TS API rd or in operation
- including security info
- ReadOpExternal:=
 - template: Set<Statement>
 - url: URL
 - time: Long
 - isTake: boolean
 - mutiple: boolean
 - securityInfo
 - operationID: String
 - transactionID: URL

- data result (triples) of a successful TS API rd or in operation to be returned to the TS API
- optionally containing a security cookie to be used by the client in subsequent requests
- DataResultExternal :=
 - result: Set<Set<Statement>>
 - operationID: String
 - time: Long
 - transactionID: URL
 - cookie: ?

- representation of error information in case an error occurred during the execution of a TS API operation
- ErrorResultExternal :=
 - operationID: String
 - errorNumber: int
 - errorDescription: String
 - cookie: ?

- enforces security policies
- checks the identity of clients and giving them the appropriate permissions
- checks TS-API and Management-API requests (external entries) and transforms them into internal entries
- transforms „outgoing“ entries into external entries for the APIs

- Client authentication
 - performing authenticity checks on clients credentials and attribute assertions
- Trust & attribute mapping
 - filtering clients supplied information (identity attributes) basing on trust relationships with asserting parties, and mapping these attributes to roles for access control
- Access control
 - taking access control decisions

- provided Entries
 - DataResultExternal
 - ErrorResultExternal
 - WriteOperation
 - write operation which has been checked by the Security Manager
 - ReadOperation
 - read operation which has been checked by the Security Manager
 - ManagementOperation
 - Management API operation which has been checked by the Security Manager

- required Entries
 - WriteOpExternal
 - ReadOpExternal
 - ManagementOpExternal
 - representation of a Management API request
 - DataResult
 - result of a TS-API operation
 - ErrorResult
 - error information in case of an erroneous termination of a TS-API operation

■ Provided

- ext-data-result, ext-err-result: to TS API, Mgmt API
- triplestore-req: to Triple Store Adapter
- mgmt-op, write-op: to Metadata Manager
- mgmt-op, read-op: to Query Processor

■ Required

- ext-read-op, ext-write-op: from TS API
- ext-mgmt-op: from Mgmt API
- triplestore-resp: from Triple Store Adapter

- representation of a Management API call
- ManagementOpExternal:=
 - operationID: String
 - operation: MgmtOperation¹
 - opParameters: List<Object>
 - securityInfo

¹ enum with all management API operations

- An authorized TS-API read operation
- ReadOperation:=
 - template: Set<Statement>
 - url: URL
 - time: Long
 - isTake: boolean
 - multiple: boolean
 - clientInfo
 - operationID:String
 - transactionID:URL
 - allowedSubspaces: Set<URL>

- An authorized TS-API write operation
- WriteOperation:=
 - triples: Set<Statement>
 - url: URL
 - time: Long
 - clientInfo
 - operationID: String
 - transactionID: URL

- An authorized Management API request
- ManagementOperation:=
 - operationID: String
 - operation: MgmtOperation
 - opParameters: List<Object>

- The result of a TS-API request
- needs to be authorized by the Security Manager
- DataResult:=
 - result: Set<Set<Statement>>
 - operationID: String
 - time: Long
 - transactionID: URL

- erroneous result of an operation
- needs to be authorized by the Security Manager
- ErrorResult: =
 - operationID:String
 - errorNumber:int
 - errorDescription: String

- provides management functionality
 - kernel startup and shutdown
 - kernel and data administration (access rights, ...)
- client invocations are translated into entries and written to the integration space
- when the operation has been performed, the API implementation reads the result and returns the result to the client

- required Entries
 - DataResultExternal
 - ErrorResultExternal
- provided Entries
 - ManagementOpExternal

- abstraction of the underlying storage infrastructure
- offers components possibility to query and store information (e.g., security policy, ...)
- consumes TriplestoreRequest entries and creates TriplestoreResponse entries of processed requests

- provided Entries
 - TriplestoreRequest
 - request to read or write triples
 - can be written by any other component
 - TriplestoreQuery
 - Request to the Query Processor to process a query
- required Entries
 - TripleStoreResponse
 - the answer to a TriplestoreRequest
 - QueryString
 - the answer to a TriplestoreQuery request

- TriplestoreRequest
 - structure needs to be specified
- TriplestoreResponse
 - structure needs to be specified
- QueryString:=
 - queryID: Long
 - queryString: String
- TriplestoreQuery:=
 - queryID: Long
 - data: Statement
 - space: URL
 - tripleSet: Set<URL>

- decomposing a query into parts that are satisfiable by the local data store and to parts that must be forwarded to other kernels
- pre- and postprocessing of queries
- reads TriplestoreQuery request from the integration space and produces QueryString

- provided Entries
 - QueryString
 - a processed triplestore query which can be executed by the Triplestore Adapter
- required Entries
 - TripleStoreQuery
 - a request to process a triplestore query

- Semantic mediation of incoming and outgoing data
- no input/output description has been provided
- possible solution:
 - reads WriteOperation and ReadOperation requests
 - creates entries with the transformed information
 - extra field in the entry determines if the entry is already in the correct format and can be processed by the other components

- creates and provides metadata
- provides routing information for the distribution manager
- reacts on Management-, Read- and WriteOperations and enriches them with (routing) information

- listening on the bus for published triples or stored structural data – create metadata.
- try to find distributed data via the semantic routing/distribution manager.
- use the query processor to retrieve the metadata already stored in the md space.
- use the storage adapter to persistently store md into a separate metadata space.
- provide a rule engine to resolve relations according to the ts ontology.

- required Entries
 - ReadOperation
 - WriteOperation
 - ManagementOperation
- provided Entries
 - ReadOperation
 - WriteOperation
 - ManagementOperation
 - The operations enriched with routing information
 - What is the structure of this routing information?

■ Provided

- StructureEntry: to Storage Adapter ... inform the storage adaptor to read/write some structural metadata to the storage
- CreatorEntry: to Storage Adapter ... inform the storage adaptor to read/write some metadata to the persistent storage

■ Required

- PublicationEntry: from ts api ... inform the mm about a published data-triple.
- ManagementEntry: from management api ... inform the mm about the creation of structural md

- handles functionality of transactions
- no input/output description has been provided
- possible solution
 - entry for each transaction API request which is answered with a entry containing the transactionID
 - on commit or rollback a entry is written to the core. Components can react on this entry if they have to compensate operations

- connects a single kernel to the global space infrastructure
- takes care of request routing and retrieval of distributed, semantically clustered data
- communicates with other kernels using the TS-API
- reacts on Read-, Write- and ManagementOperations which have to be routed to another kernel and writes the answer to the integration space

- required Entries
 - WriteOperation
 - ReadOperation
 - ManagementOperation
 - RoutingInformation
 - required information to route the request
 - should this be part or the Operation request?
- provided Entries
 - DataResult
 - ErrorResult
 - dist-kernel, dist-descr, dist-rel
 - what is this?

Component-Component Matrix



to from	Triple Space API	Management API	Triple Store Adapter	Security Manager	Mediation Manager	Metadata Manager	Query Processor	Transaction Manager	Distribution Manager
Triple Space API			Triplestore Request	WriteOpExternal ReadOpExternal					
Management API			Triplestore Request	ManagementOpExternal					
Triple Store Adapter	Triplestore Response	Triplestore Response		Triplestore Response	Triplestore Response	Triplestore Response	Triplestore Query Triplestore Response	Triplestore Response	Triplestore Response
Security Manager	DataResultExternal, ErrorResultExternal	DataResultExternal, ErrorResultExternal	Triplestore Request			ManagementOperation, WriteOperation, ReadOperation			
Mediation Manager			Triplestore Request						
Metadata Manager			Triplestore Request						WriteOperation, ReadOperation, ManagementOperation, RoutingInformation
Query Processor			QueryString, Triplestore Request	DataResult, ErrorResult					WriteOperation, ReadOperation
Transaction Manager			Triplestore Request						
Distribution Manager			Triplestore Request	DataResult ErrorResult		dist-kernel, dist-descr, dist-rel			

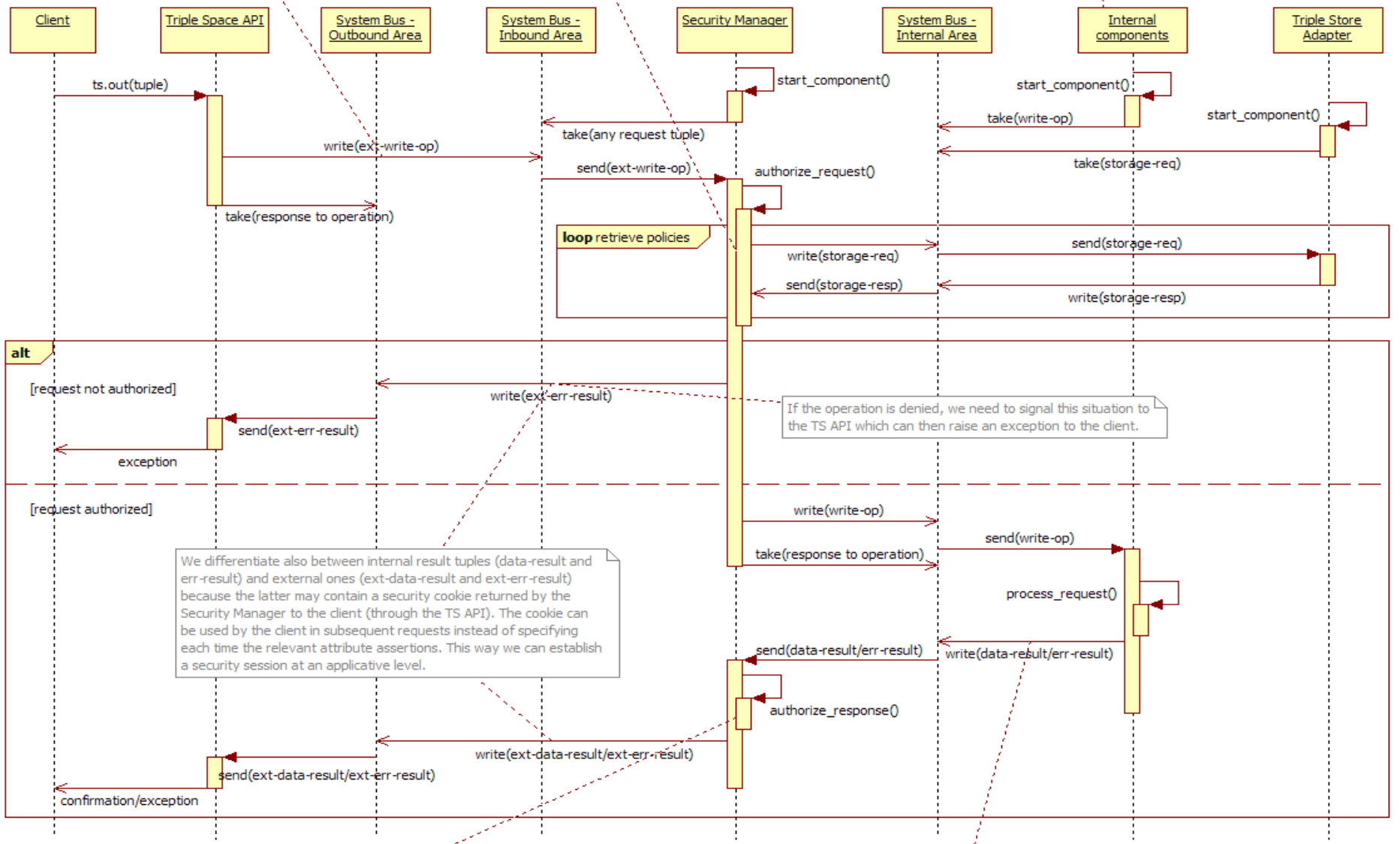
Security Manager sequence diagram for OUT operation

(the diagram is similar to the ones for RD and Management operations)

We differentiate among external operation (ext-write-op, ext-read-op, ext-mgmt-op) and internal ones (write-op, read-op, mgmt-op) because the firsts may carry one or more assertions or a security cookie.

The number of policies retrieved depends on the hierarchy of spaces accessed by the request. In the case these retrieval operation will reveal to be a bottleneck, we will consider the introduction of a caching mechanism internal to the Sec. Manager.

Note: from the point of view of the Security Manager, it doesn't matter which internal component will take up the tuple published in the internal area. It may be one or multiple components. If possible, we prefer the solution where we publish only a tuple in a single, internal, general-purpose space. Alternatively, we can publish several tuples each one in a separated internal space (personally I don't like this solution, but for us it would be ok).



We differentiate also between internal result tuples (data-result and err-result) and external ones (ext-data-result and ext-err-result) because the latter may contain a security cookie returned by the Security Manager to the client (through the TS API). The cookie can be used by the client in subsequent requests instead of specifying each time the relevant attribute assertions. This way we can establish a security session at an applicative level.

If the operation is denied, we need to signal this situation to the TS API which can then raise an exception to the client.