OCF overview

DANA team (i2CAT)

OFELIA Control Framework origin

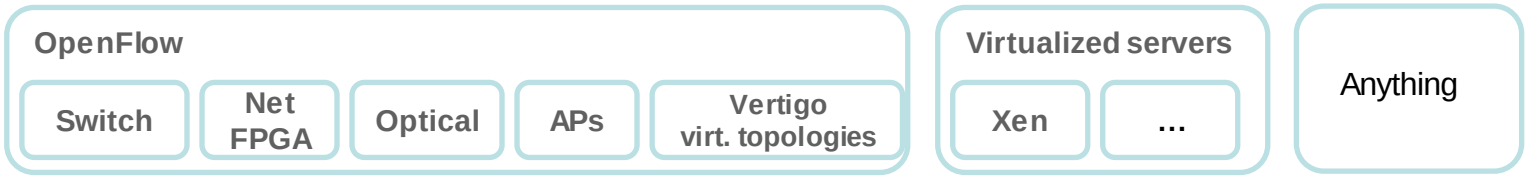
- Born within the OFELIA project
- Extended within multiple projects
  - FIBRE
  - ALIEN
  - FELIX
  - Fed4FIRE
  - Géant3+
  - Others…

OFELIA Control Framework (OCF)

- What is?
  - OCF is an open source management software originally developed to be used in the OFELIA facility
- What does it offer?
  - Scalable and distributed architecture
  - Resource life-cycle management
  - Resource monitoring
  - Automatic experiment orchestration using distributed and heterogeneous resources
  - Resource independence through the use of AMs with AMsoil (later)

OCF: capabilities

- Supported resources



- Extensible to new resources: AM base class (AMsoil)
- Easy federation schema:
  - Federation at AM's level: isolating administrative domains
  - Intra federation out of the box
  - Inter federation focused on resource sharing and possible through multiple interfaces (SFA, GENI…)
- Multiple user interfaces
  - GUI
  - CLI
  - …

OCF: technologies

Front-end

Back-end: general purpose

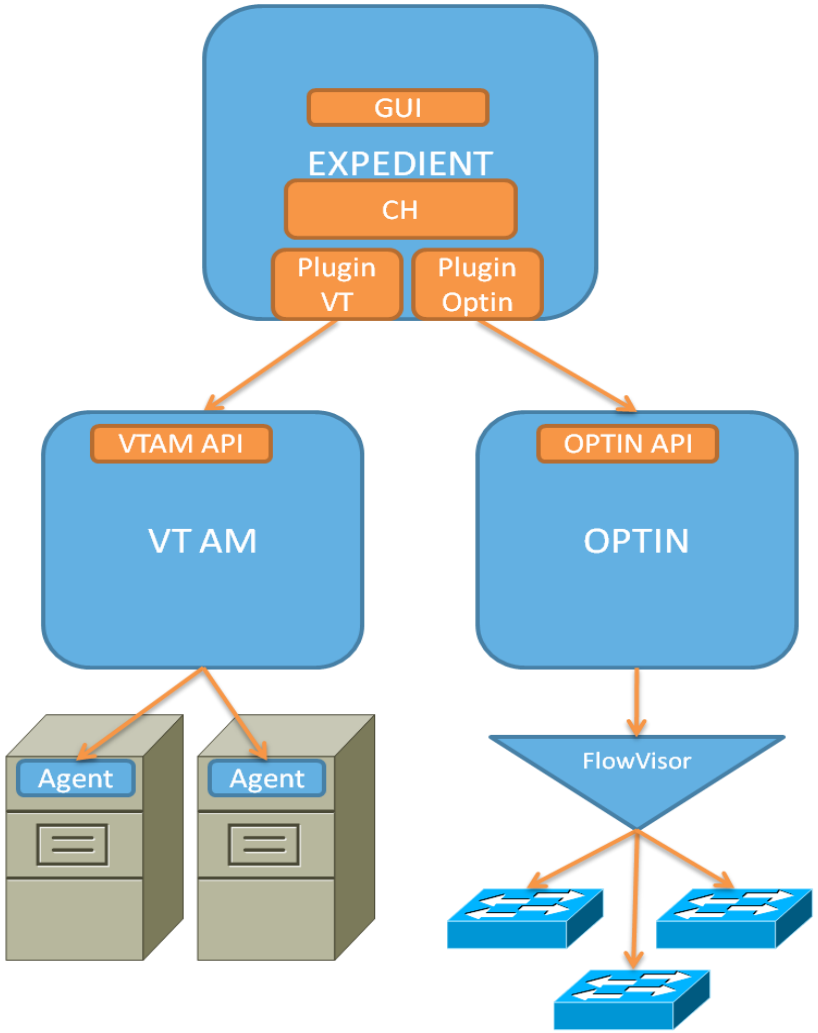
Backend: virtualization

Infrastructure: OpenFlow

OCF: status

- Under constant work
- 0.x versions
  - Based on Stanford's Expedient (only UI+CH) and opt-in
  - Support for OpenFlow 1.0 equipment, XEN virtualized servers
  - Basic monitoring on AM connections and VMs status
- 1.x versions
  - New architecture: APIs, AAA, RSpecs, AMs
  - Complete monitoring and provisioning flows
  - Opt-in manager replaced by FOAM
  - Isolation of clearinghouse and driver-based UI
  - AMs/RMs provided with a Policy Engine (rule based policies on resources)
  - Integration of VeRTIGO & VT planner

Basic OCF Components

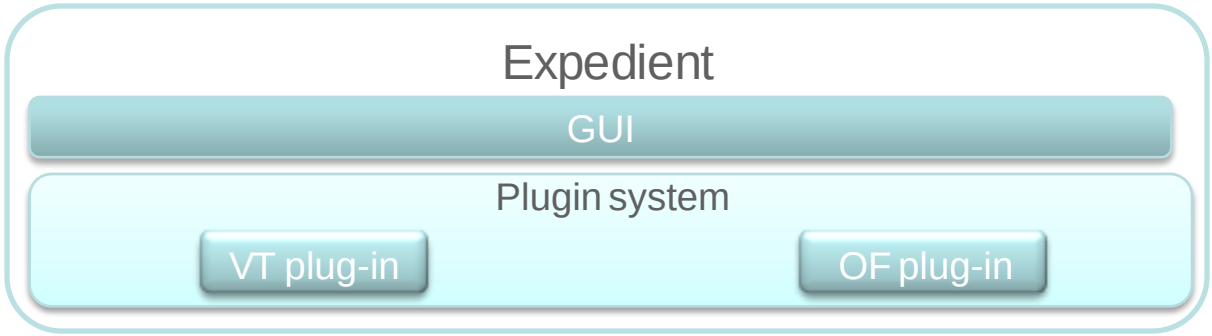


- Expedient:
  - Admin/Experiment web UI
  - Clearinghouse: projects/slice management, user permissions
  - AM plug-ins: handling visualization, resource specific communication
- VT AM: virtualized servers AM
  - API: XMLRPC, custom RSpec
  - Agent: hypervisor management
- Opt-in Manager: OF AM
  - API: GENI XMLRPC, OF RSpec v1
  - FlowVisor: proxy, slice FlowSpace multiplexor

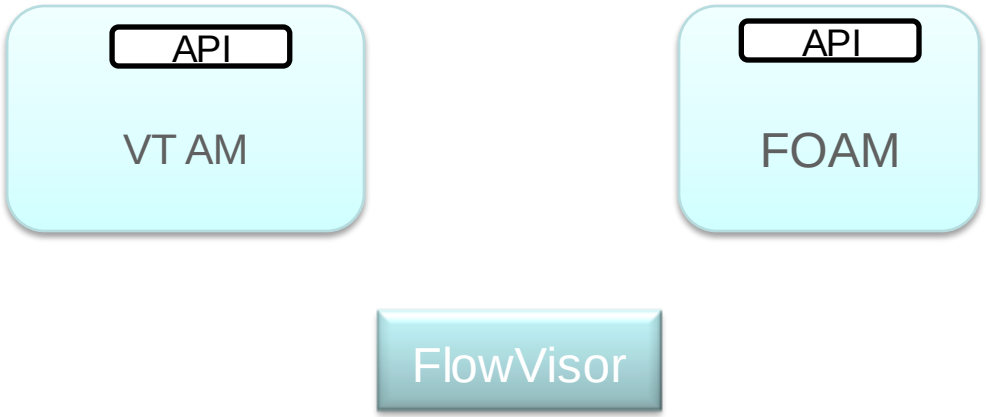
Aggregate Managers

- Entities that manage resources of a specific type (e.g. VMs, switches) within a project
- Virtualization Aggregate Manager
  - The VT AM controls the virtualization resources of the testbed (virtual machines within the servers)
  - Experimenter can create VMs with desired characteristics
- OpenFlow Aggregate Manager
  - The OF AM controls the OF resources of the test bed (OF-enabled switches)
  - Experimenter can define FlowSpaces involving these and the VMs (based on IP or mac address, port, VLAN, etc.) and set a controller which updates the flow tables of the switches

v0.5 architecture: layered system

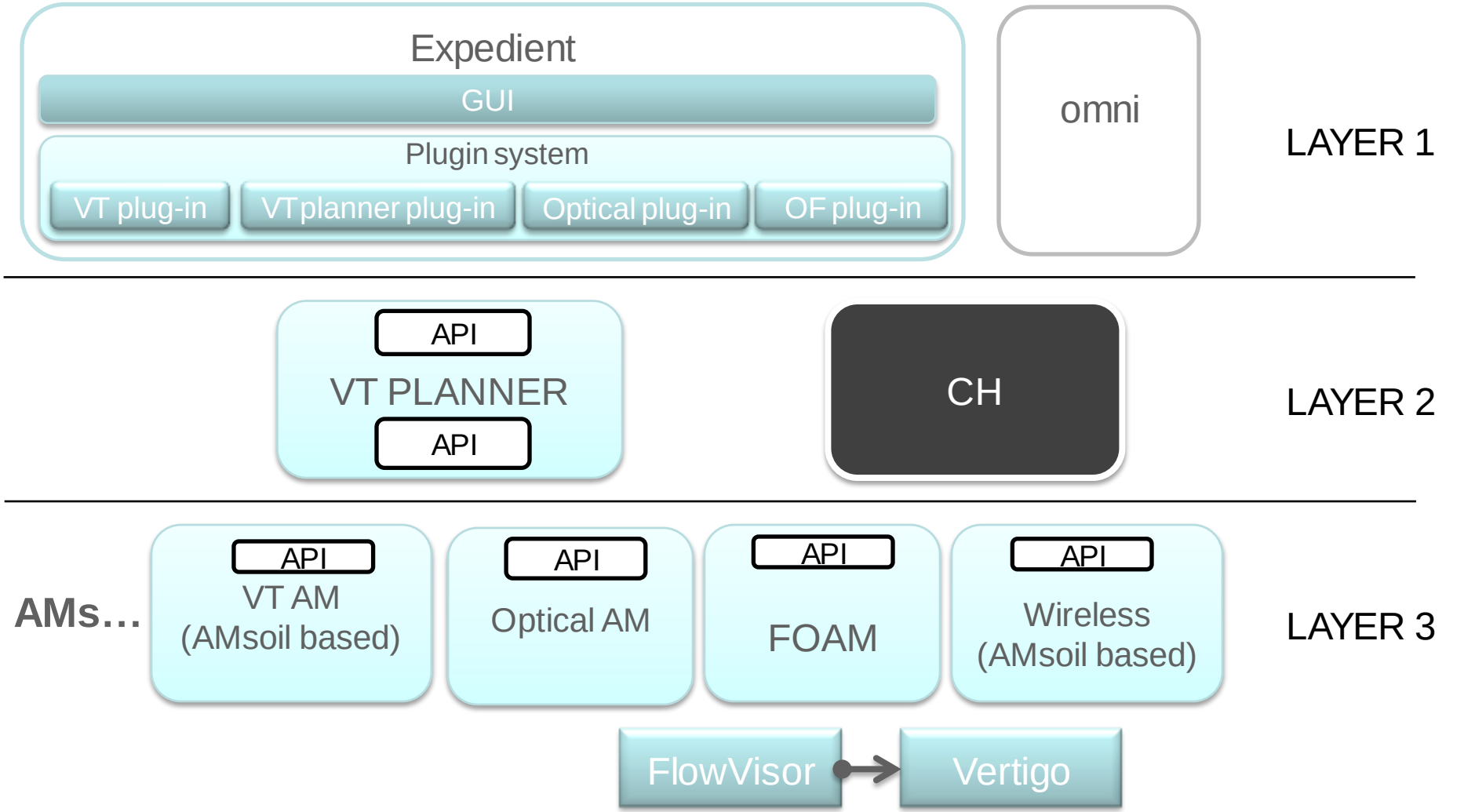


LAYER 1



LAYER 2

v1.0 architecture: layered system

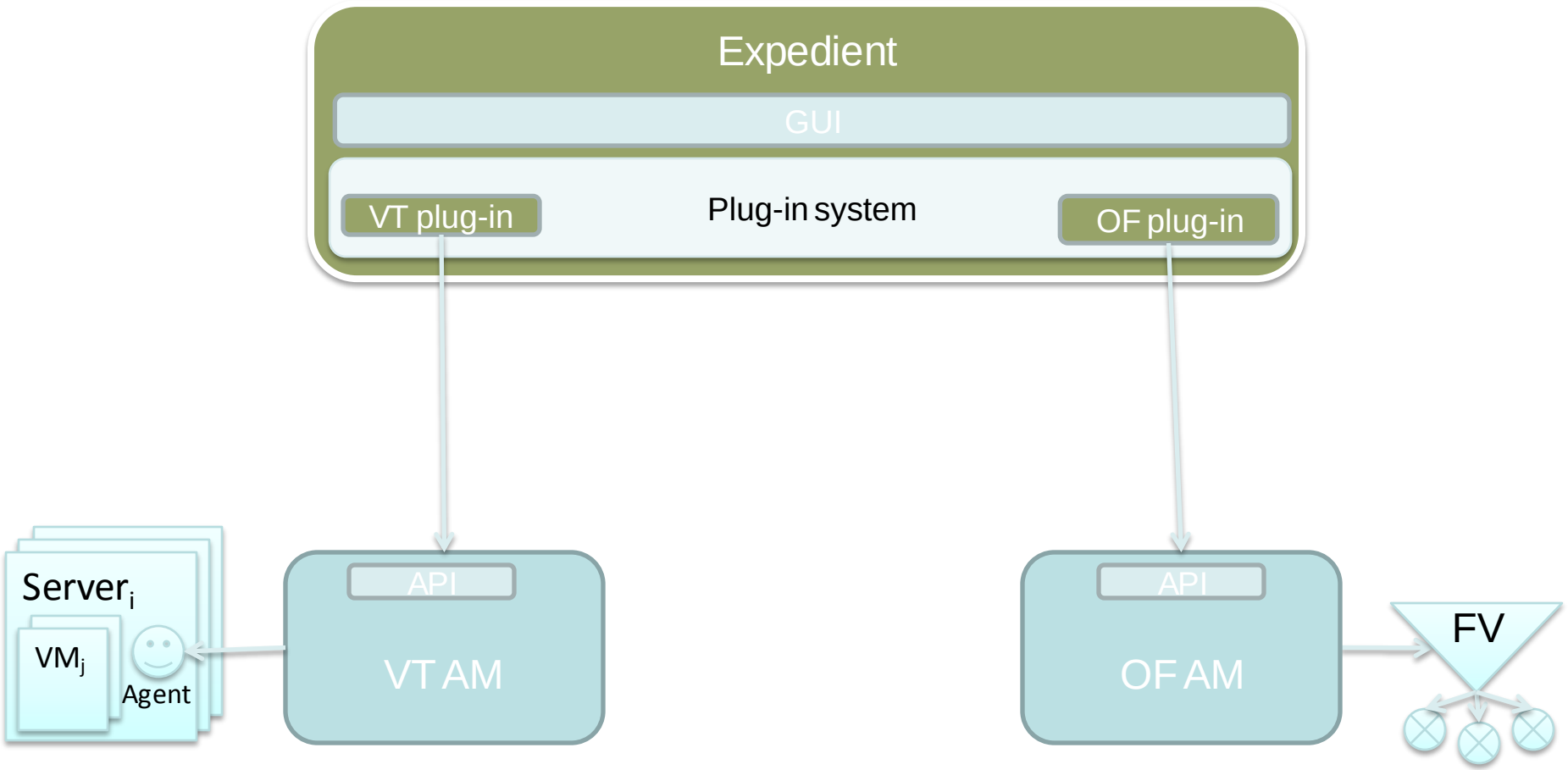


v0.5 architecture: plug-in system (I/II)

- Concept of plug-in
  - Convenient way to expand capabilities of an application
- Why is it useful for OCF?
  - < 0.5: to develop a new AM implied fiddling with Expedient internals
  - ≥ 0.5: AM developer needs not to worry about modifying the core
- Developing a plug-in
  1. Create folder under plug-ins path. Name this after the plug-in
  2. Appropriately fill settings. You may also add your own
  3. Define method `get_ui_data(slice)`:
     - Input: slice. May be useful to filter data or return to given slice page
     - Output: dictionary with (at least) keys “nodes” and “links”
       - “Node” and “Link” are wrappers around some basic info
  4. Set URLs to access plug-in methods from UI (e.g. add AM/related resource)

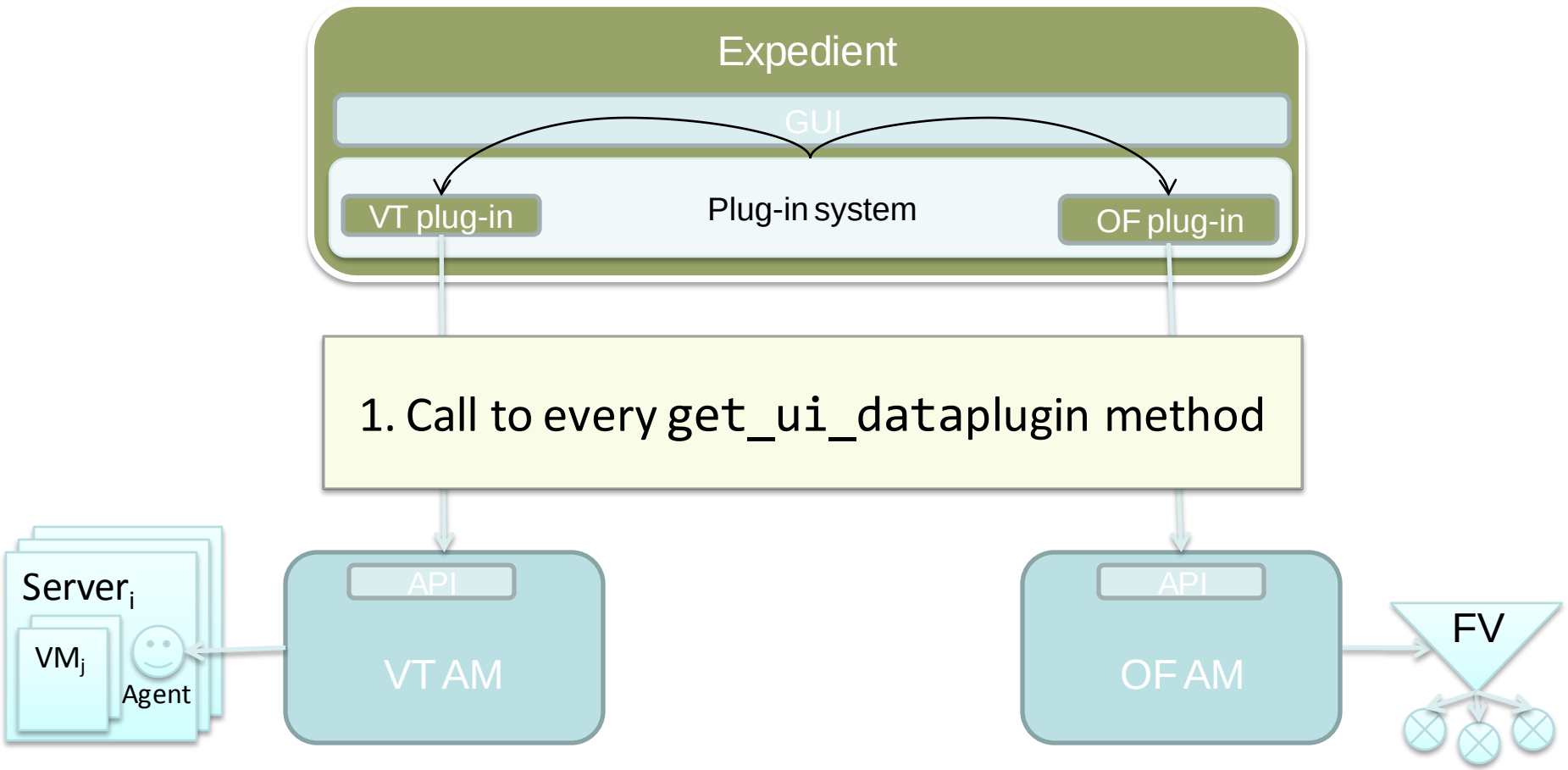
v0.5 architecture: plug-in system (II/II)

- Interaction Expedient – plug-in – AM_i



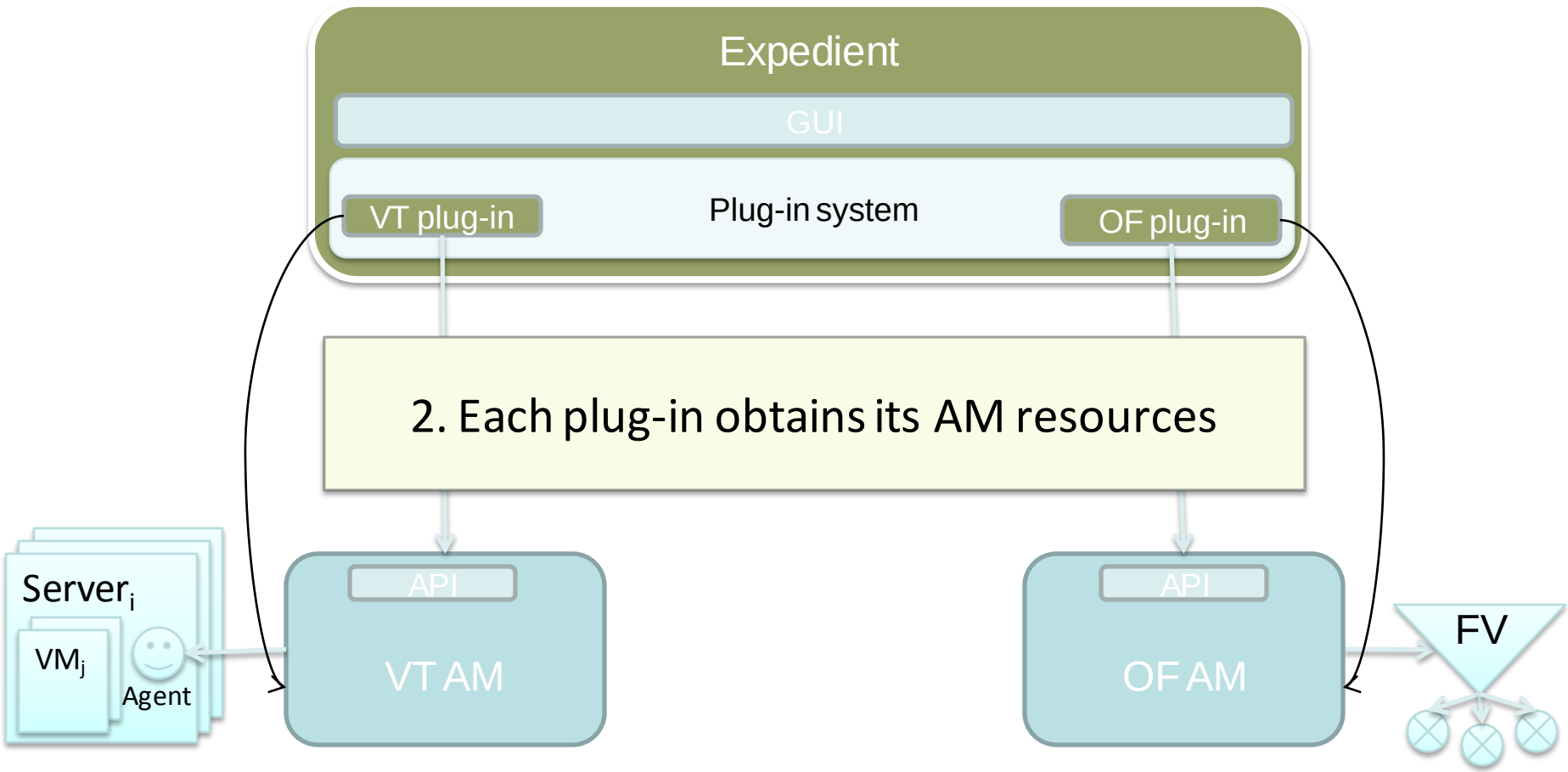
v0.5 architecture: plug-in system (II/II)

- Interaction Expedient – plug-in – AM_i



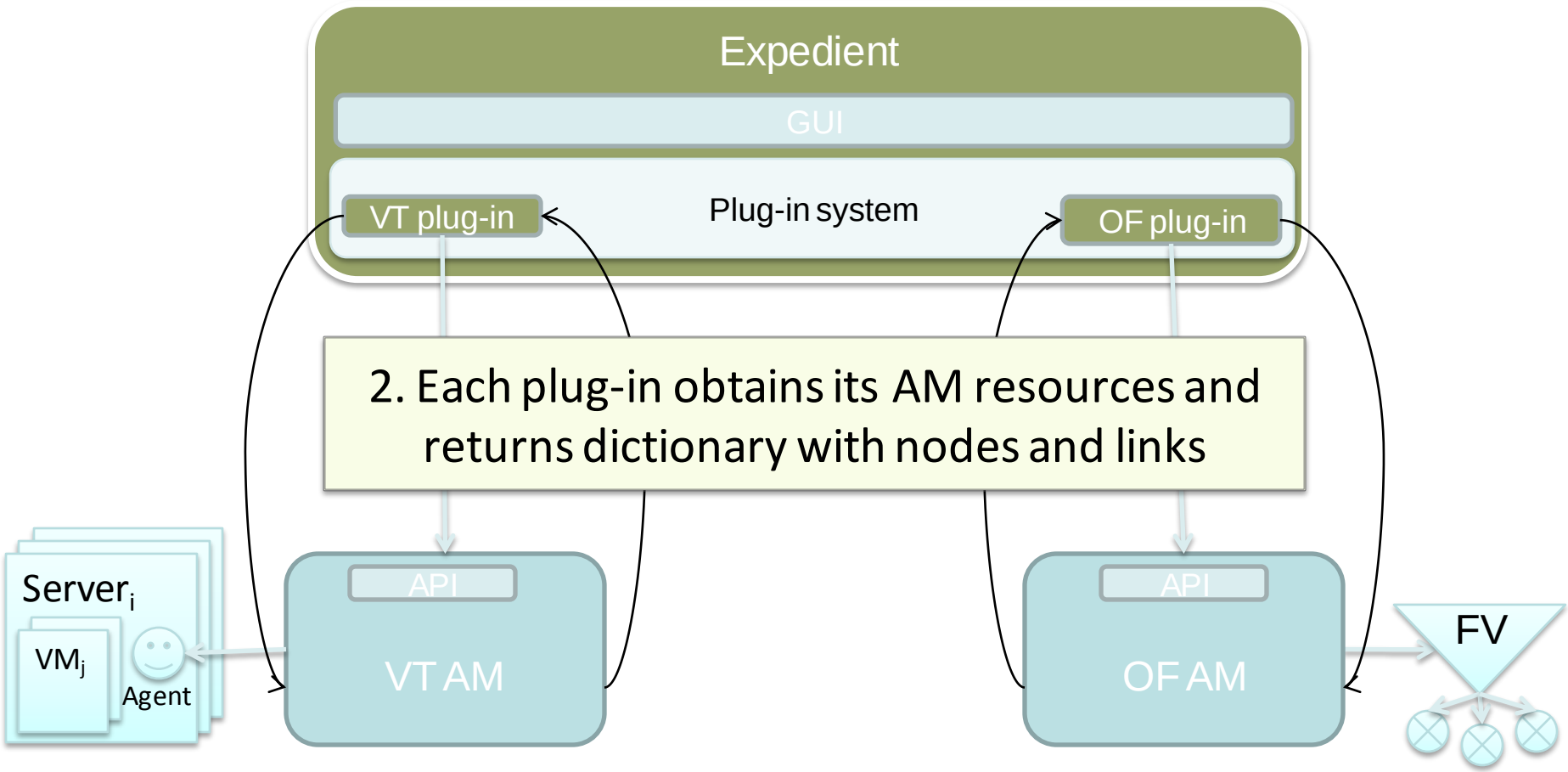
v0.5 architecture: plug-in system (II/II)

- Interaction Expedient – plug-in – AM_i



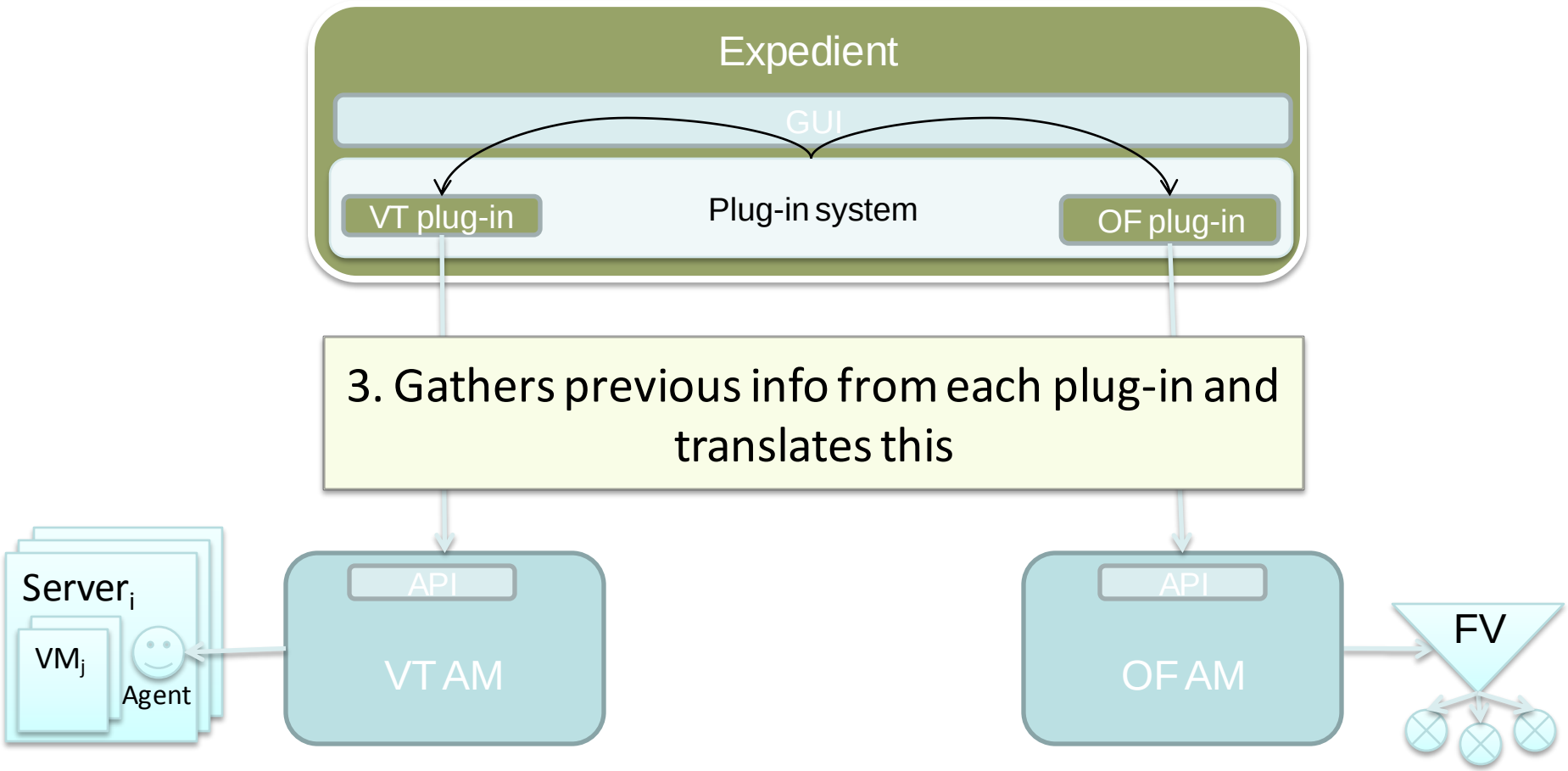
v0.5 architecture: plug-in system (II/II)

- Interaction Expedient – plug-in – AM_i



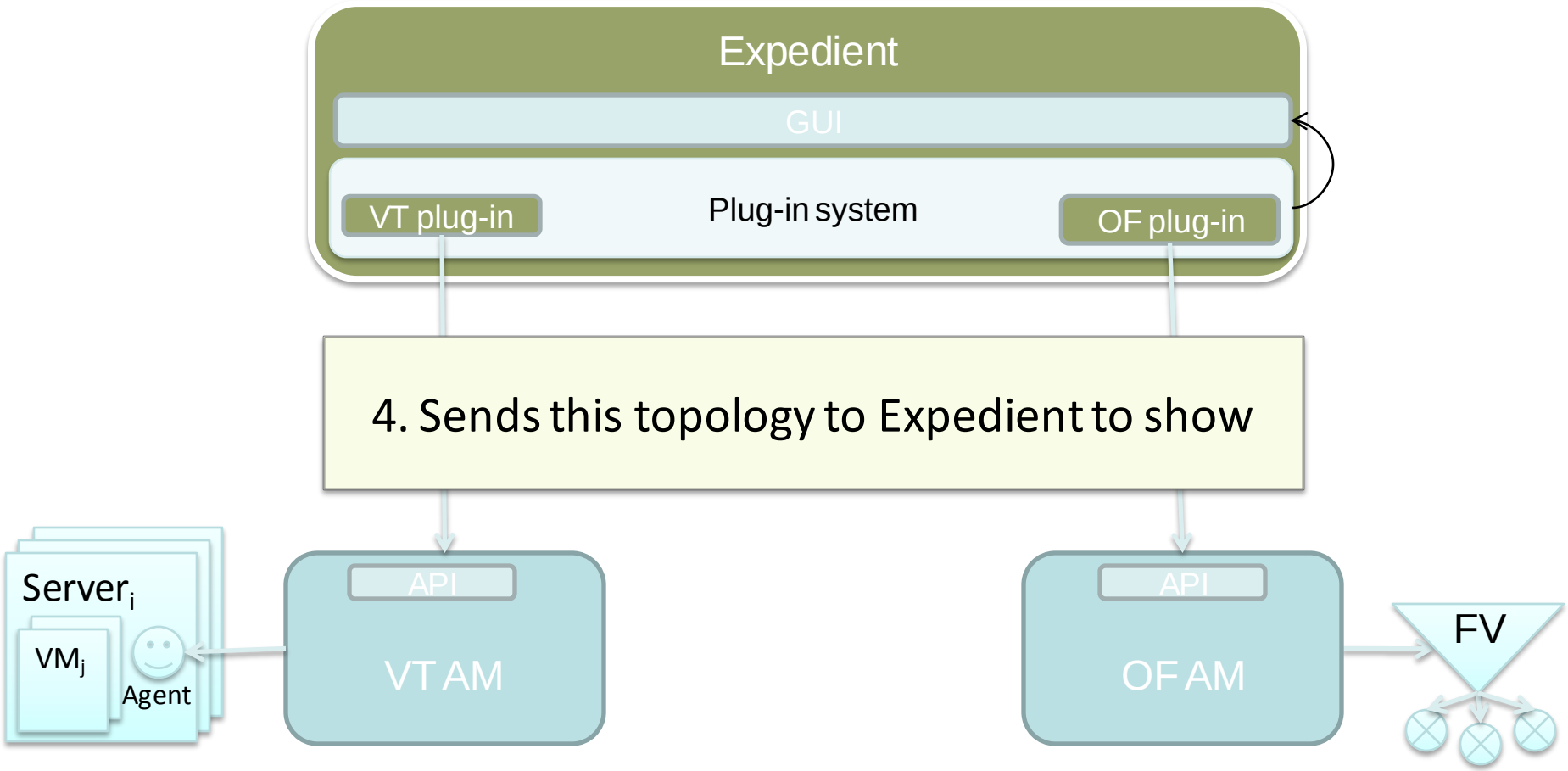
v0.5 architecture: plug-in system (II/II)

- Interaction Expedient – plug-in – AM_i



v0.5 architecture: plug-in system (II/II)

- Interaction Expedient – plug-in – AM_i

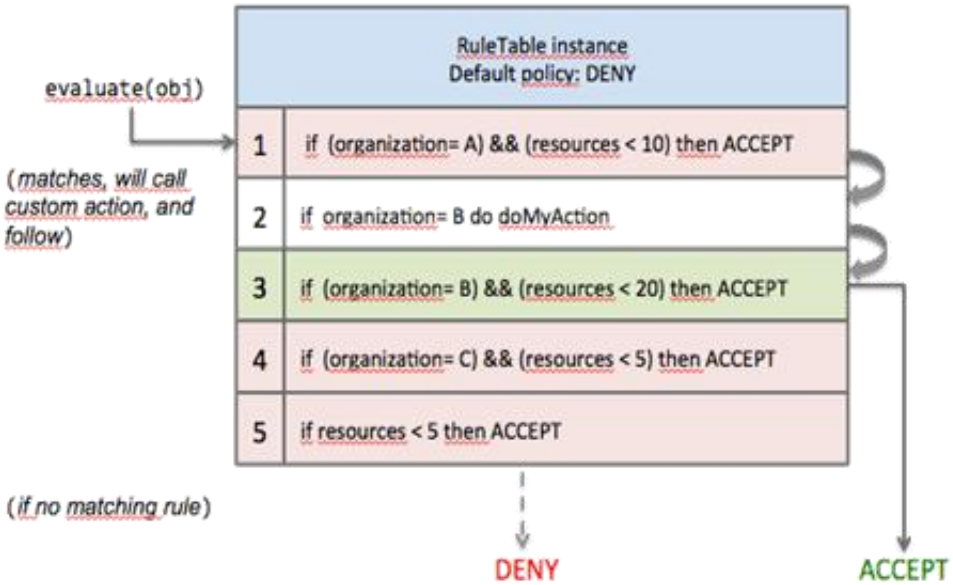


pyPElib: what is it? (I/V)

- Small library that helps programmers by using abstractions
- This allows to build rule-based **Policy Engine(s)** within a certain scope of action
- Similar idea as in IPtables
  - *https://github.com/fp7-ofelia/pypelib*
- Supports:
  - Rule-based policy enforcement
  - Rule-based action triggering, logging
  - Multiple scope support within an application
  - Multi-syntax support
  - Flexible syntax customization (Resolver and mappings)
  - Multiple persistence back-ends

pyPElib: RuleTable (II/V)

- Encapsulates a set of policies of a certain scope
- Result of an evaluation will always be a True/False value
- Composed of:
  - Set of rules
  - Default policy (ACCEPT/DENY) in case no rules match



- evaluate() method receives an object which encapsulates the basic information to be checked (requests, RSpecs…)
- The information is verified against the ruleset in order
- Rules may interact with other classes/functions or modules of the application to obtain extra information

pyPElib: Rule (III/V)

- Internal objects used to **represent a specific policy**
  - **Terminal** rules: if condition matches, break rule lookup loop and **return** rule's **ACCEPT / DENY** value
  - **Non-terminal** rules (action rules): if condition matches, they **do actions** without returning any value
- Conditions can be simple or complex, containing sub-conditions
- Can be **modified**, **reordered** or **deleted** on the fly
- If required, **persisted** using a backend
- Rule form:

  if <[not] condition> then <return_value> [nonterminal] [do <action>] [denyMessage <error_message>] [#<rule_comment>]

  Examples:

  if vm.memory>256 then DENY denyMessage Memory limit is 256MB

  if request.user in collection {user1,user2,user3} then deny do log # Deny requests from user 1, 2 and 3

pyPElib: mapping (IV/V)

- Contains the basic association between keywords and objects, functions or static values
- Defined by the user of the library according to the rules needs
- Condition mappings

```
condMappings = {
    "server.name": "metaObj.server.get_name()",
    "server.uuid": "metaObj.server.get_uuid()",
    "server.status": "metaObj.server.get_status()",
    "number.vms": ControllerMappings.getNumberOfVMs,
    "vm.name": "metaObj.server.virtual_machines[0].get_name()",
}
```

- Action mappings

```
actionMappings = {
     "None": "None",
     "LogVM": ControllerMappings.logVM,
}
```

pyPElib: mapping objects & methods (V/V)

- metaObj
  - Request file (RSpec, XML, SFA…) parsed and “objectified”
  - You can get attributes from the metaobject

- Define methods to get needed values or perform actions:

```
@staticmethod
defgetNumberOfVMs(metaObj):
    from vt_manager.models import VirtualMachine
projectUUID = str(metaObj.server.virtual_machines[0].get_project_id())
    return len(VirtualMachine.objects.filter(projectId = projectUUID))

@staticmethod
deflogVM(metaObj):
ControllerMappings.logger.debug("[NAME:%s|RAM:%s|HD:%s]"
  %(eval("metaObj.server.virtual_machines[0].get_name()"),ControllerMapping
  s.getVMMemory(metaObj),ControllerMappings.getVMHDMemory(metaObj)))
/getValueFromConfiguration
```

OFVER: OFELIA versioning system (I/II)

- Library to help with installation and upgrade
  - Alternative to software bundling
  - Downloads code directly from repository
  - 1 OFVER per OCF module (e.g.: expedient, vt_manager…)
  - Blocking: only 1 installation per module at the same time
  - *https://code.google.com/p/ofver/*
- Location
  - `/opt/ofelia/<module>/bin/ofver`
- How to use
  - `./ofver {install, upgrade} [-{d , n , b , f}]`

OFVER: OFELIA versioning system (II/II)

- Procedure
  1. Initialize modules and settings
  2. Recover upgrade status (if any)
  3. Obtain code from repository
     - `git pull` under `/opt/ofelia`, `/opt/ofelia/oxa/repository` (XEN server)
  4. Check local version number
  5. Move through version hook files (if any – otherwise fall to default hooks)
     1. pre-{install, upgrade}-hook
     2. {install, upgrade}
     3. post-{install, upgrade}-hook
  6. Update local version number

END