DIANOMIC

Introducing FogLAMP Manager

The Industry 4.0 Edge Management System

FogLAMP Manager is a management tool for scaling and securing an Industry 4.0 edge. Whilst FogLAMP can be completely managed via its own API and User Interface, FogLAMP Manager offers a number of key features and benefits over the management of individual FogLAMP instances.

Benefit	**Feature**
Easily scale up and out	Centralized management of all FogLAMPs
Enables many users w/ different skill sets	Self-Serve multi-tenancy for users and administrators
Understand your edge data pipelines physically and logically	Abstract FogLAMP components
Security, monitor remote assets, reliability	Manage FogLAMP instances not routable from central location
Fast easy rollback if problems occur, security, reliability	Version control for all FogLAMP configuration data
Continuous health check of entire edge data path	Centralized monitoring of all FogLAMP instances
Fast, easy scale out of similar assets, data sources, integrations and applications	Templating of FogLAMP configurations and application pipelines
Rollbacks, compliance, security	Change management with audit and logging

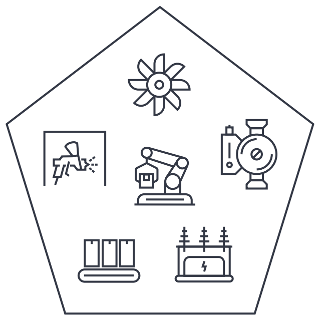
It is also important to understand what FogLAMP Manager is not:

- FogLAMP Manager is not in the data path for FogLAMP data collection. Data does not pass through FogLAMP Manager ever.
- FogLAMP Manager does not provide any mechanism to view the data collected by individual FogLAMP instances.
- FogLAMP Manager does not manage the operating system upon which FogLAMP is running.
- FogLAMP Manager does not manage the hardware upon which FogLAMP is running.

The remainder of this document will provide a brief overview of each of the key features listed above and introduce the concepts behind FogLAMP Manager. This document is not intended to replace the product documentation but rather give the flavor of FogLAMP Manager's features and functions.

Industry 4.0 Edge Physical Components

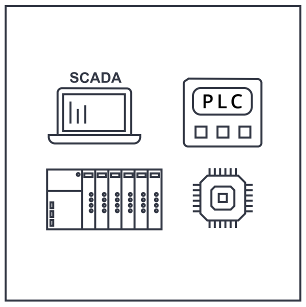
Assets



Object(s) being monitored

- Fans
- Pumps
- Transformers
- Floors
- Paint booths
- Robots
- etc.

Source Devices



Monitors assets and creates data

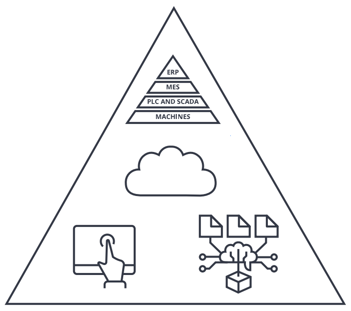
- SCADA, PLC, DCS
- Aggregating head end system
- Physical proxy for sensors
- Multiplexer connecting sensors
- Sensors
- Sometimes: Source Device embedded in Asset

Edge Device

Runs FogLAMP

- Gateway
- VM
- Container
- Sometimes: Edge Device embedded in Asset, Source Device or External System

Integrations

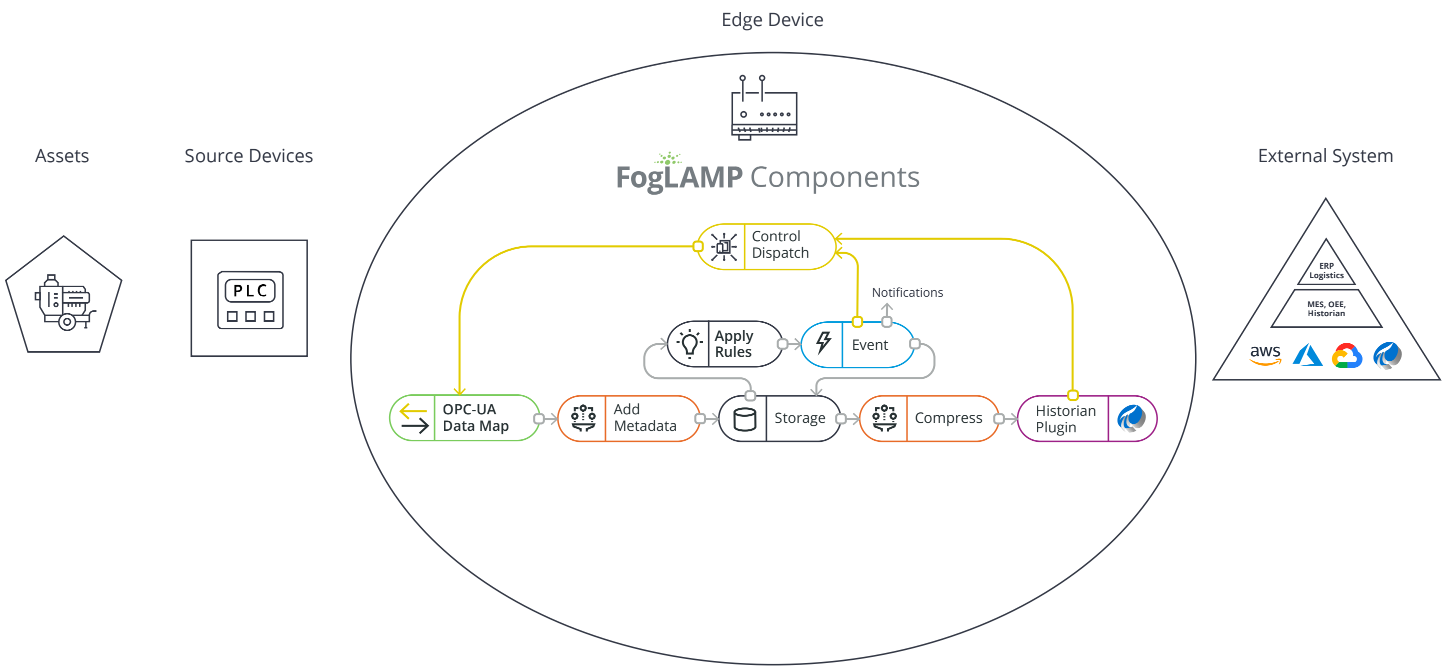


Destination to export data from FogLAMPs:

- ISA95 System
- Cloud
- ML System
- HMI
- Any

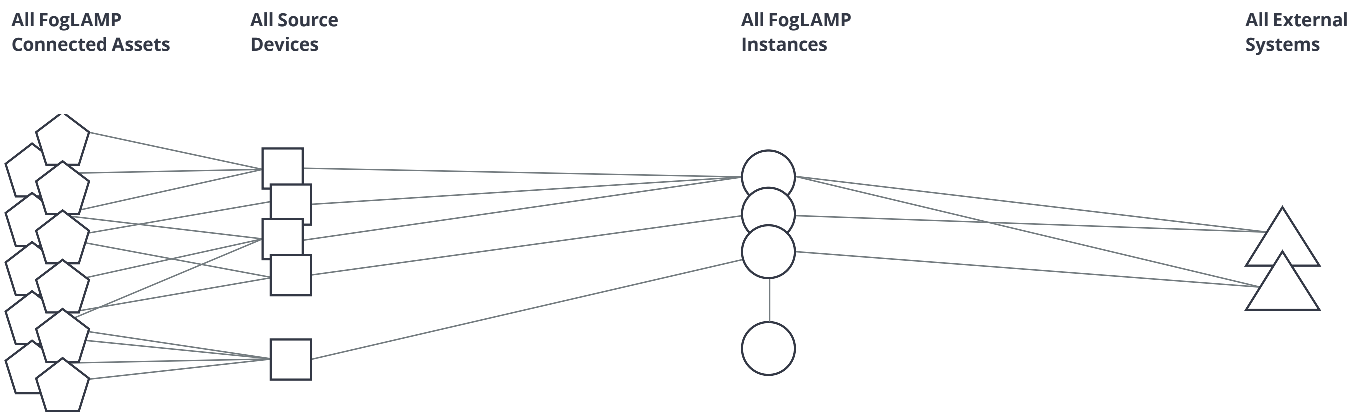
DIANOMIC

Basic FogLAMP pipelines include: Data Ingest, Ingress Filters, Storage, Rules, Notifications, Events, Egress Filters And Data Stream Integrations. Bi-directional control can be used for non real-time control.



FogLAMP
Manage

Scale, secure, and manage Industry 4.0 Edge



Centralized Management

Centralized management is the primary feature of FogLAMP Manager, providing a multi-tenant point from which large numbers of FogLAMP instances can be administered by data scientists, engineers, operators, maintenance and business users with IT oversight, audit and compliance. Whilst feasible to administer a single FogLAMP or even a small group of FogLAMPs using the FogLAMP user interface, the limitations of such an approach soon become apparent:

Switching becomes multiple screens
- Duplication of data
- Centralized security
- Synchronization of changes across connected sets of FogLAMPs
- No single point of truth for system wide configuration and status
- The need to deal with each individual instances for version management
- Multiple backups to manage

FogLAMP Manager addresses these limitations whilst also adding system level scale, control and audit features for an enterprise wide 4.0 Edge. There are some features of FogLAMP Manager that may be deemed important enough that managing a single FogLAMP instance with FogLAMP Manager becomes worthwhile.

Single Screen

FogLAMP manager allows a large number of FogLAMP instances to be managed using a single application and user interface to configure and manage those instances, there is no longer a need to connect to each instance in turn to update the instance or to verify the performance of the FogLAMP instance.

A single sign on to the FogLAMP manager provides all that is needed to manage all instances, FogLAMP manager itself will keep track of the individual credentials required for each FogLAMP which it manages.

Groups of FogLAMP instances may be created to make it easier to manage those FogLAMPs that are logically related to each other. A single FogLAMP may exist within multiple groups.

Configuration Duplication

FogLAMP Manager provides a number of mechanisms that allow the administrator to reduce the amount of configuration data that is duplicated. This may be accomplished by use of the data abstraction model, templates or through the use of macros. The administrator can merely update a value in one single location within FogLAMP Manager and have it propagate to all the FogLAMP instances rather than go to each FogLAMP in turn and set the value. Thus the process is not only quicker and easier it also reduces the risk of errors or missing updates to some of the FogLAMP instances. This becomes particularly important if a FogLAMP instance is unreachable at the time the administrator attempts the update.

Security

As part of the process of taking an existing FogLAMP instance into the control of FogLAMP Manager, or adding a new FogLAMP instance, the manager will apply a security profile to the FogLAMP instance. That security profile will enforce encryption of the FogLAMP REST API and also convert the administration account to use certificate based authentication, with the certificates being managed by FogLAMP manager.

FogLAMP manager also controls all account creation within the FogLAMP, resulting in tighter central control of the authentication credentials with restricted local access to the FogLAMP instance. This restriction on local access can be relaxed from the central FogLAMP Manager, however FogLAMP Manager is always considered the holder of the definition of truth. Any changes made locally in the FogLAMP will be overridden by the central management unless action is taken from the FogLAMP Manager.

Since FogLAMP Manager maintains inventory of all software deployed and centrally manages updates, deletes and rollbacks it can reconcile common vulnerability and exposures (CVEs) and be used to deploy software updates that contain security patches.

FogLAMP Manager templates (see below) enable hashing of compliant application components and complete applications. Compliant components and applications can then be reused and tracked by the organization enabling a self-serve compliant 4.0 Edge.

Change Synchronization

The FogLAMP Manager uses an interaction style in which an administrator prepares a set of changes for one or more FogLAMP instances and then deploys to all of those instances in a single operation. This results in the administrator being able to create very complex changes that affect multiple instances and having those changes applied within close proximity to each other. FogLAMP Manager does not however provide a model in which all updates must be accepted across a set of FogLAMP instances before any instance actions the update.

Options exist to also allow a single FogLAMP to be updated or a subset of FogLAMPs to be updated in a single operation using the grouping facility of a FogLAMP.

FogLAMP manager can also perform automatic retries of deployments to FogLAMP instances that are not reachable at the time the deployment request is made. This can be important when large numbers of FogLAMPs are being managed as it allows updates to occur for the majority when one or more instances have issues.

Single Point of Truth

Since FogLAMP Manager has the complete set of software versions and configurations for each FogLAMP node it manages, it provides a means to ensure the entire fleet of FogLAMPs are running the current version of the software components and the correct configuration. Should a node be compromised or have local updates performed upon it, the central FogLAMP Manage can be used to restore that FogLAMP to the known and correct state.

Version Management

FogLAMP Manager offers a version control system for all configuration information across the set of FogLAMP instances that it manages. This allows each deployed configuration to be version controlled, with the ability to roll back to a previous version at any time.

Configuration updates are first staged in a new version and then, once complete for all FogLAMPs, that version is deployed to the physical FogLAMP devices. At this time, the version is locked and no further changes can be made to that version. If changes are required then a new version should be created using this version as the base configuration.

Alternatively a previous version can be used as the base version, allowing for modified versions of previous deployments to be run. This locking behavior, whilst it may seem prescriptive, allows the previous version to be installed in the knowledge that no updates will have occurred to that version since the last time that version was running.

Backup/Disaster Recovery

FogLAMP Manager holds the complete set of configuration data for each FogLAMP it manages. This in effect makes FogLAMP manager a complete backup for all the configuration data of all your FogLAMP instances. A single FogLAMP may backup its own data, both configuration and buffered device data, however that backup is run on the local FogLAMP and resides on that FogLAMP.

FogLAMP Manager holds the full configuration data and is able to restore a FogLAMP back to a known state if it becomes corrupted, local configuration changes have been made or in the case of a complete failure a new piece of hardware can be provisioned and commissioned as a replacement. Clearly any buffered device data will be lost as FogLAMP Manager does not have access to this.

Abstraction

FogLAMP Manager provides an abstracted model of the components in a FogLAMP instance rather than merely reflecting the physical software components and their configuration. This allows for two very important features:

- The real world components are more visible.
- Data can be shared between multiple FogLAMP configurations reducing duplication.

Abstract Model

The abstract model of the FogLAMP world that FogLAMP Manager has is probably closer to the actual components that you would find in the real world, whilst being an abstraction of how FogLAMP itself views the world. This is probably easier to explain if we look at a simple FogLAMP implementation, in this case a single FogLAMP that connects to a FLIR camera and sends data to an OSIsoft PI Server.

If we first look at a FogLAMP view of this configuration we have:

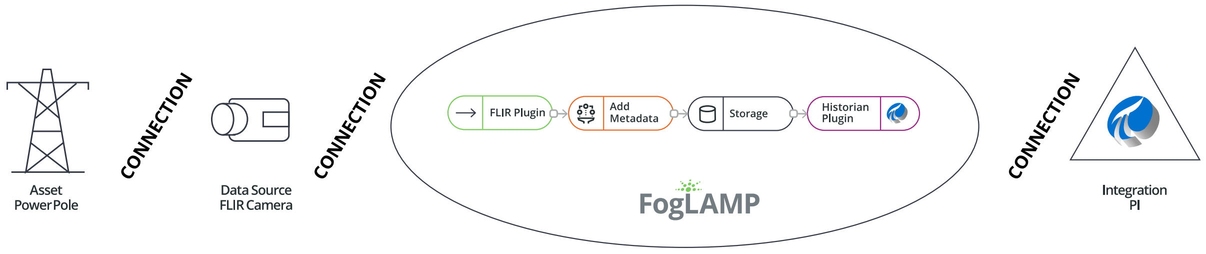
- A south service with the FLIR plugin loaded. This service is configured with all the configuration data it needs to pull the thermal data from the FLIR camera; IP address and authentication token.
- A north task with the OMF plugin loaded. This task is configured with all the configuration data needed to communicate with the PI WEB API endpoint of the PI server. This includes the authentication data, IP address, AF structure, etc.

The FogLAMP Manager view of this same setup however is very different. In FogLAMP Manager what we actually have is four distinct systems:

- A FogLAMP
- An “integration” which represents the PI Server
- A “data source” which represents the FLIR camera
- An “asset” which is the item of equipment we are monitoring with the FLIR camera

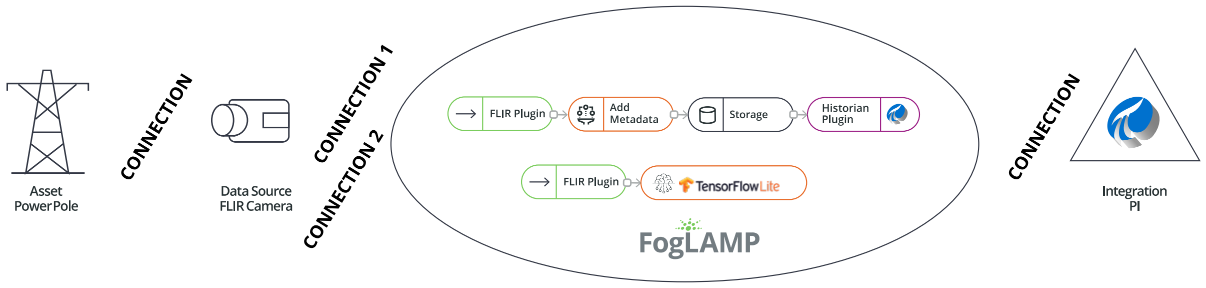
We then have three connections; one from the asset to the FLIR camera, one from the FLIR camera to the FogLAMP and another from the FogLAMP to the PI Server.

The configuration of the system is now split across the seven elements rather than simply the two services/tasks as in the FogLAMP view. Although this might seem more complex at first sight, it allows the configuration data to be attached to the real world entities that define the values. An example of this is the address and credentials for the PI Server can be associated with the integration that represents the PI Server.



Whilst this may not seem of particular importance in this single instance of FogLAMP it allows for better data sharing where multiple FogLAMPs are involved or when machines or instances are used by multiple services within a FogLAMP.

To illustrate this, with our single FogLAMP instance we will add a second south service to the FogLAMP. This service will load an image processing machine learning plugin that will connect to the same FLIR camera and take the visual data feed from the camera. In the FogLAMP model of things the new south plugin would require a set of configuration information to allow it to connect to the camera.



In the FogLAMP Manager abstract view however, we merely create a new connection between the FogLAMP and the FLIR camera. The difference between this connection and the original connection is the type of that connection, one is a connection for thermal data and the other a connection for the image processing model. Since the IP address of the camera is associated with the camera itself and not the connection, this means we do not need to enter the camera's IP address a second time. Moreover if we decide to change the IP address of the camera we simply change the IP address of the camera once.

The same is true of the connection to the PI Server, we put the IP address and credentials of the PI Server with the integration and then any FogLAMP that wishes to connect to that PI Server can reuse that data. If we had 50 FogLAMP's all connected to the same PI Server, sharing the same credentials, then we only have to enter them once and when we change the credentials we only have to do it in one place.

Configuration Data Placement

The data associated with these abstract entities maps onto data needed by the south and north plugins of FogLAMP, and how the data for a particular plugin is split between the entities depends mostly on where it naturally belongs. When FogLAMP Manager creates the FogLAMP configuration it will collect data from the abstract entities to create the configuration for the plugins.

South plugins for example will tend to have some data collected from the asset entity,some from the data source entity and some from the connections. Typically the asset will have the data that describes the particular asset whilst the connection will have the data that describes how to connect to the asset. For example the asset would have things like the IP address of the asset, any protocol map for the asset and asset metadata. The connection would have information like the rate to poll data from the asset, any processing to perform on the data from this particular asset as it relates to this connection to FogLAMP.

In the above description we have assumed that we have an intelligent asset from which we can directly collect data. However we may also have assets that have no intelligence, in which case we have an external sensor, or maybe several, that are monitoring the asset. These are data sources in the FogLAMP Manage abstract model. An example of this would be our FLIR camera from above, the asset might be an item of electrical switch gear, the camera would be the data source with a connection between the switch gear and the data source to represent the fact the camera is pointing at the switch gear. Information about the switch gear would be applied to the asset, whilst camera information would be applied to the camera. The connection from the camera to the FogLAMP would have data regarding the processing specific to the data from that camera to the given FogLAMP.

We could also have a hybrid configuration in which we have an intelligent asset that has a direct connection to a FogLAMP and also has a second method of collecting data not available from the asset itself via some external sensor.

Similarly with integrations they would typically have data that describes the system while the connection between FogLAMP and the integration would describe how the data is sent or connection specific information. In the case of a PI Server the address of the PI Server would be with the integration, whereas the location on the AF Structure would be with the connection as each connection may have a different requirement for this.

These are no hard and fast rules, some implementations may have different requirements regarding what relates to an integration, asset or data source and what relates to a connection. These can be tailored on a per implementation basis by use of custom templates.

Non Routable FogLAMP Instances

It is a fairly common situation that a FogLAMP instance is installed in such a network topology that it is not possible for an administrator to always be able to connect to a FogLAMP instance to manage it. This may be because the FogLAMP is behind a firewall or the FogLAMP instance may not always have network connectivity; for example if it is on a mobile asset or has power restrictions.

FogLAMP Manager allows for the management of those FogLAMP instances by the use of a local agent that runs as a microservice alongside the FogLAMP. This agent initiates calls from the FogLAMP to the centralized FogLAMP Manager in order to push status information to FogLAMP Manager and to pull configuration updates to FogLAMP.

The agents may be configured to make these calls to the centralized FogLAMP Manager based on a schedule or on some local event, such as network connectivity or connection to a docking station in the case of a mobile asset.

The result is that a central administrator is able to update the configuration of an otherwise unreachable FogLAMP instance and have that update actioned at the first available opportunity. That centralized administrator is also able to collect performance data from the FogLAMP when the opportunity occurs.

Version Control

FogLAMP Manager's configuration management includes the concept of version control as the basis for all configuration changes. This allows:

- All configuration changes to be tracked, producing an audit trail of changes.
- Versions of configuration are locked when they are deployed such that they can not be altered after deployment.
- A deployment to always be returned to a previous point in time.

The version control model is such that changes can only be made in versions that are not locked, as soon as a version is deployed it is locked and can not then be modified. When a new version is deployed the previous version is not unlocked, this gives the assurance that if, at some later time, it is decided to roll back to this previously deployed version, it remains as it was when it was last used.

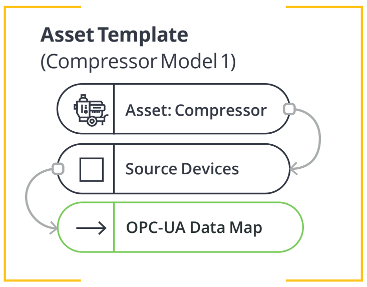
Centralized Monitoring

FogLAMP Manager provides a single location to determine the current state of each FogLAMP it manages and to observe the traffic flow rates through those FogLAMPs.

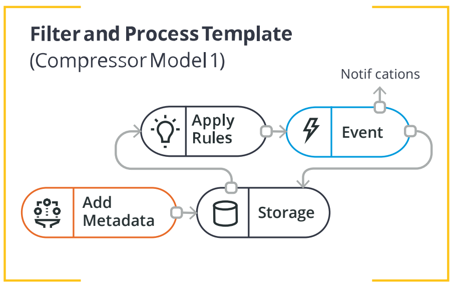
Templating Configuration & Applications

FogLAMP Manager makes extensive use of templates as the way to define how objects are created within the management framework. Templates are used to represent the skeleton objects within the system, such as external systems, connections and machines. They may also be used to represent collections of objects.

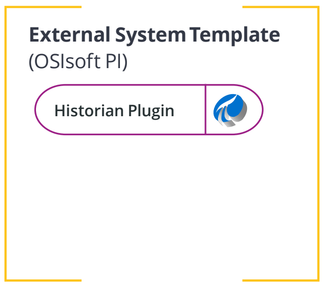
The templating system is designed such that individual implementations of FogLAMP Manager can generate templates that are tailored to the requirements of that implementation. The templates can define default values that can be used to customize the behavior of an object.



A reusable blueprint mapping the data registers to a specif c type of asset.



A reusable data transformation blueprint for a specif c type of asset.



A reusable method and data mapping blueprint to an External System.

Template Types

FogLAMP Manager supports a number of different templates types:

- **Asset:** This template type describes the items being monitored in the logical model that is manipulated by the FogLAMP Manager.
- **Connection:** This type of template describes how elements in the logical model are connected together.
- **Data Source:** A data source represents external sensors or data collection devices.
- **Integration:** An integration template models the systems north of FogLAMP that receive the data from FogLAMP. This may be the cloud services or the on premise data historians into which data is processed from FogLAMP.
- **Filter:** A filter template is a base template for defining a single filter that can be applied to a connection. It defines the processing elements that may be applied to the data as it traverses the connection.
- **Process:** A process template provides an order set of filters that may be applied as a single entity to the data that traverses the connection.
- **Notification:** The notification template provides the template for defining the rules to evaluate on the data and the mechanism for delivering notifications when those rules trigger.
- **Collection:** A collection template is a template that combines multiple templates to allow single templates to be created that will define a complete tree of items to be created. Each template within the collection may be of any of the template types, thus allowing machines, connections and processes to be combined. A collection template may also include collection templates, allowing hierarchies to be created.

An Asset Template Example

A simple example of this would be a template for connecting to a Modbus device. If using the FogLAMP modbus plugin for this you must provide a map that is used to map specific modbus registers to assets. Using the template mechanism, and assuming a modbus controlled pump as an example, you can create a template for your pump that has a default map for that pump. You can now call this template the XYZPump. Whenever you want to add one of these pumps into your FogLAMP system you can merely create a new pump, as a machine, using the XYZPump as the template for that machine. Adding a new pump does not require any map to be given as this now comes from the template.

Not all configuration items need to be set in the template, thus when you create a new XYZPump you will still be asked to provide the information that is not set in the template, for example the pump name and the IP address of its modbus controller.

If at some later stage you wish to add another asset which maps to some other modbus registers you merely create a new version in FogLAMP Manager and update the XYZPump template to include that new asset. Now deploy that new version to all the FogLAMPs and every pump defined from this template will now have the extra asset mapped from the modbus.

Filter Pipeline

No code application development is a feature of FogLAMP that is implemented by taking a series of processing filters and applying these filters to a data flow either at the south during ingestion or in the north during egress. Within FogLAMP Manager these filters are applied to the connections between machines and FogLAMPs or between FogLAMPs and external systems.

Filter pipelines are also templated in FogLAMP Manager, allowing templates to be written that define the entire process flow, again with parameters that may be requested on creation of a pipeline instance from the template.

These filter pipeline templates can again be created dynamically by the administrator of a particular FogLAMP Manager deployment, thus allowing application templates that are tailored to a particular business need.

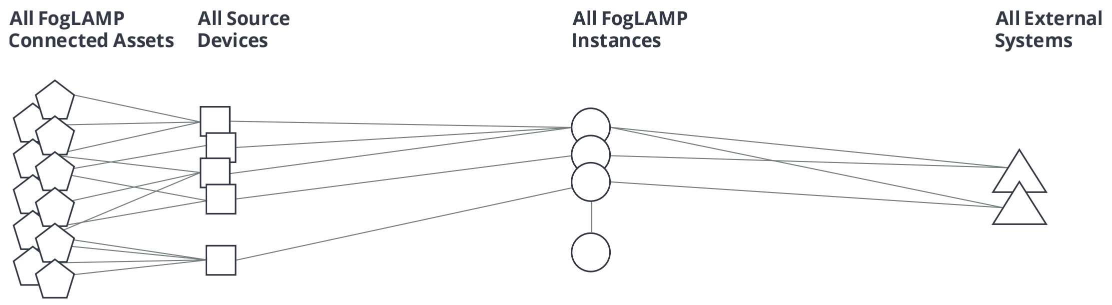
Filter templates allow complete tailored applications to be created and reused across multiple FogLAMP instances, with version control and consistency of implementation guaranteed by FogLAMP Manager.

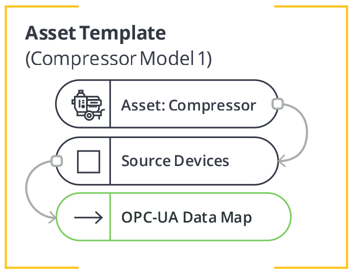
Notification & Event Templates

Another powerful feature of FogLAMP is the ability to implement actions on the edge using customized rules and delivery mechanisms. These too may be templated within FogLAMP Manager to apply consistent rules and actions using the same versioning mechanism across multiple FogLAMP instances.

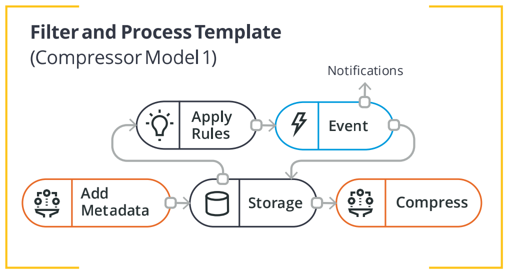
Templates of Templates

The template mechanism in FogLAMP Manager is not just limited to creating individual entities such as assets, data sources, connections, FogLAMPs, processing pipelines and integrations; the mechanism may also be used to create complex trees of objects, with each of the elements in the tree consisting of entity built from templates of their own.

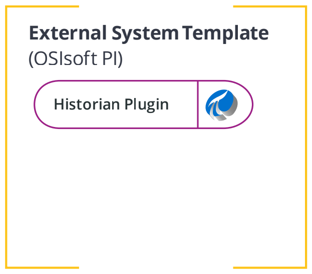




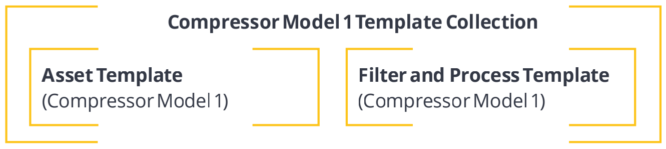
A reusable blueprint mapping the data registers to a specific type of asset.



A reusable data transformation blueprint for a specific type of asset.



A reusable method and data mapping blueprint to an External System.

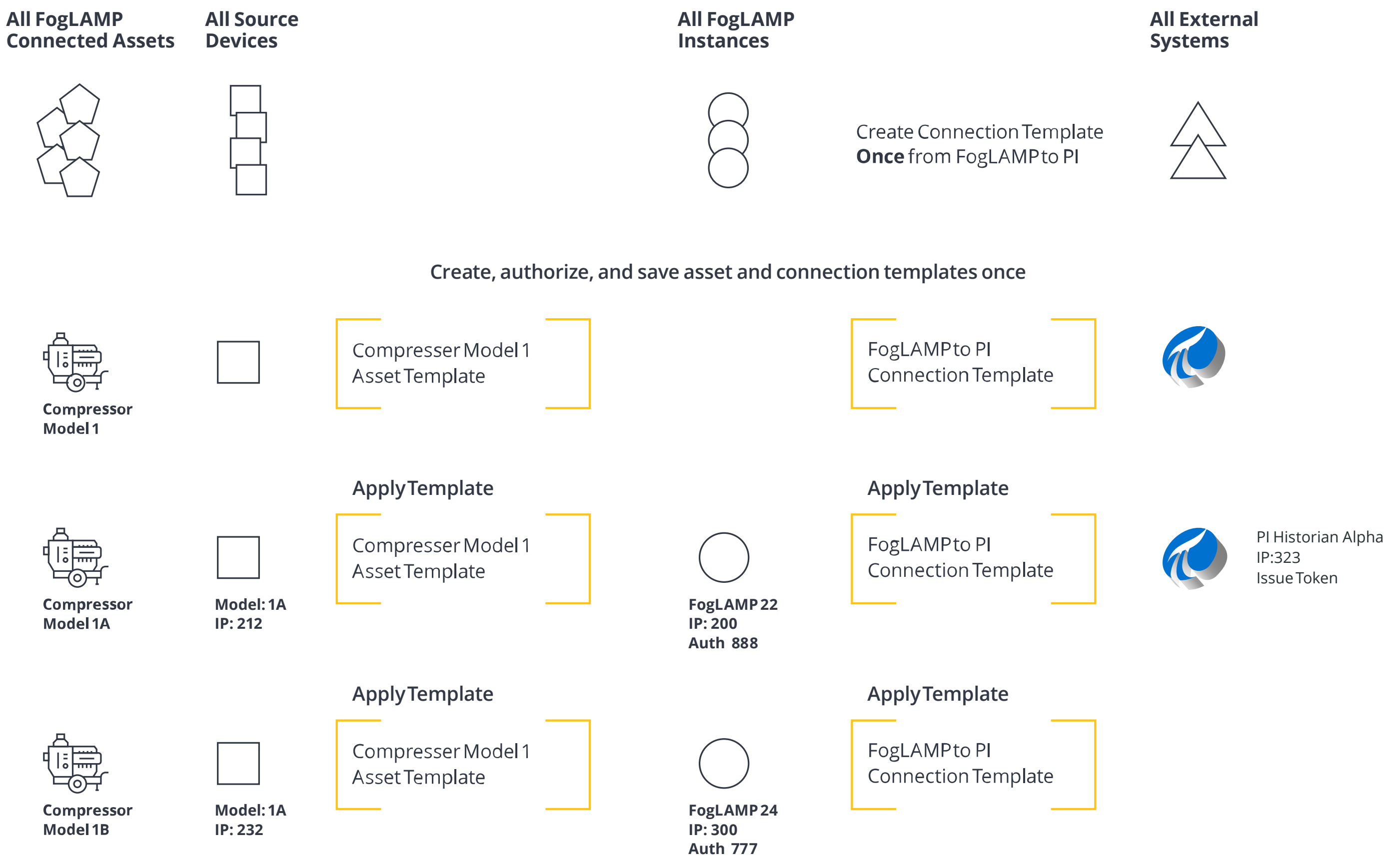

Compressor Model 1 Template Collection
Asset Template
(Compressor Model 1)
Filter and Process Template
(Compressor Model 1)

A reusable collection of multiple templates to create no code data pipelines.

For example, if a complex item of plant required many connections to it with different processing pipelines on each, this could be built from a hierarchy of templates:

- The item of equipment itself would have a template that defines the item and the properties of that item.
- Each data source that is used with the asset would have a template of its own.
- A template would be created for each connection type required between that plant, data sources and the FogLAMP with the filter pipelines required.
- A further template would then be added that would define the set of templates that would then be applied to each asset to FogLAMP flow.

Applying this template would create an asset, prompting for the asset instance properties and connect it, via the complete set of connections and data sources to FogLAMP to which it was applied.


All FogLAMP Connected Assets
All Source Devices
All FogLAMP Instances
All External Systems
Create Connection Template **Once** from FogLAMP to PI
Create, authorize, and save asset and connection templates once
Compressor Model 1
Compresser Model 1 Asset Template
FogLAMP to PI Connection Template
Apply Template
Apply Template
Compressor Model 1A
Model: 1A IP: 212
Compresser Model 1 Asset Template
FogLAMP 22 IP: 200 Auth 888
FogLAMP to PI Connection Template
PI Historian Alpha IP:323 Issue Token
Apply Template
Apply Template
Compressor Model 1B
Model: 1A IP: 232
Compresser Model 1 Asset Template
FogLAMP 24 IP: 300 Auth 777
FogLAMP to PI Connection Template