# Proposal: Gregor von Laszewski

Version faas 0.1

# Function as a Service

Most recently function as a service has become an important addition to cloud computing services. It represents an architectural pattern which introduces an abstraction allowing the users to not focus on the issues related to managing infrastructures in the cloud. Instead it allows the user to define functions that can be invoked by the users without thinking about the complexity of setting up an execution infrastructure.

## History and current implementations

The history and current implementations include:

* <http://hook.io/> (2014)
* AWS Lambda <https://aws.amazon.com/lambda/> (Jul. 2015 ?)
* Google Cloud Functions
* Microsoft Azure Functions
* OpenWhisk
* OpenFaas

Although some of the services promote very short functions, OpenFaas allows longer running functions which are especially attractive for many complex Big Data applications as we have identified in our use cases.

To enable an FaaS integration into the BDRA we need to think about how we can

1. Specify functions
2. Deployment of functions
3. Register functions
4. Discover functions

This section will provide the necessary specification and abstraction to achieve each of the tasks

## Function Registry

A function registry allows us to add functions with their specification, their implementation and potentially their reproduction in general terms of the infrastructure on which such functions are deployed.

Figure 1: Function Registry

Authorization

Service

Authentication

Service

Operations

Function Repository

Client

Function   
API

Figure 2: Example Architectural Function Registry Pattern

Clients have to be able to execute two different tasks

1. Uploading a function to the registry
2. Discover a function to be executed in the registry.

Thus, we need to first specify a function and use our example based specification for this.

Let us assume *spec* I represents the specification of a function. We need to upload that specification to the registry. How we specify a function we will describe in Section ??.

The Registry is defined as follows

### Registry

Object ??: Registry

{

"registry": {

“endpoint”: <http://cloudmesh.org/faas/registry>, # just an example link

"function": [ "objectid:function" ]

“querryinterface”: “specifies which query interface is used, .e.g. SQL, MongoDb, …”

}

}

### Operations

Operations that need to be supported by a FaaS Registry include

* Upload: add a new function to the registry
* Delete: delete a function from the registry
* List: list the functions
* Search: search for functions based on a search query. The query format is determined by specifying a query interface that could represent the backend of where the function registry data is stored.
* Get: retrieves a function specification

## Function Specification

Specification of the functions must be straight forward and allow the promotion of reuse across different function registries and “providers”.

Description of the parameters

{

"Type" : "The type of the Function",

"Properties" : {

“Description”: {

"Name" : “The name of the function”,

"About" : “The description of the function”,

“Tags”: [“a list of tags to identify the Function”],

}

“Environment”: {

"Runtime" : [“Runtime parameters to be set (python, java versions, …)”],

"Variables" : “[list of environment variables]”,

"MemorySize" : “Maximum memory used by the function in MB”,

"Timeout" : “time in seconds”,

}

“Specification”: {

"Code" : “Code”,

“Language”: “The language of the code (python/java/go/…”,

"ExecutionHandler" : “The handler that is executed when the function is called”,

"FailureHandler" : “Behavior based on Failure of the Function”,

“LogHandler”: “A handler that can be specified to lof the execution”,

}

Security: {

“EncryptionInput”: [“Type of encryption used for each input parameter”],

“EncryptionOutput”: [“Type of encryption used for each output parameter”],

"Role" : “When executing the function it is executed in this Role”,

}

“Properties”: {

“Parameter A”: “additional properties that may enable additional features”

}

}

## Security

The function definition has been developed with a minimal set of security in mind that includes authorization, authentication, as well as encryption on input and output parameters. The later is flexible enough to allow encryption on selected input and output parameters. If no encryption is chosen for the parameter “None” is specified.