XTAO Bioflow

REST API Specification

v1.3

WWW.XTAOTECH.COM

Exploit Data Potential

XTAO Bioflow 1
REST API Specification 1
v1.3 1
Overview 3
Authorization Header 3
Backend Management 4
Backend Task Notification 8
Pipeline Item Management 10
Pipeline Management 20
Job Management 35
Appendix 1. Python Version Sample Code 77
Appendix 2. Golang Version Sample Code 80
Appendix 3. Java Version Sample Code 83

Overview

The Bioflow REST API v1.3 defines the APIs between client and Bioflow server, the APIs include 6 parts:

- Authorization Header
- Backend Management;
- Backend Task Notification;
- Pipeline Item Management;
- Pipeline Management
- Job Management
- User Management

Authorization Header

All users of Bioflow should apply for API key and Security key from administrator, User would use the two keys to build HTTP Authorization Header of Restful API. Authorization header can validate user who send the request and make sure request message is valid. Any invalid request would be denied.

How to build Authorization Header?

Prerequisite:

1. Get information of user who is sending request.
   a) User name. (mandatory)
   b) UID (optional)
   c) Group name (optional)
   d) Gid (optional)
   e) Umask (optional)
2. Get user's key pair from administrator:
   a) API key: ***akey***
   b) Security key: ***skey***

Six Steps to build Authorization header:

1. Encapsulate user information to a JSON format string.

   {

       “user”: “xiaoming”,

       “group”: ”xtao”,

       “uid”: “1001”,

       “gid”: “1001” ,

       “umask”: ”0022”

   }
2. If skey is more than 16 characters, assign ***skey*** to dkey directly. If ***skey*** string is less than 16 characters, padding skey to 16 characters and assign it to dkey. IE.

   dkey := fmt.Sprintf(“%16s”,skey)
3. Encrypt user information JSON string to ahead by AES algorithm with dkey, and assign ahead to a new HTTP header “**X-XTAO-Account**”
4. Generate an upper case string by all headers of the HTTP request: Method\n\Host\nURI\nHeader1\nHeader2\n…
5. Calculate signature from the upper case string by HMAC/SHA256 algorithm and skey
6. Append HTTP “**Authorization**” Header and assign it to “APIKey=***akey***, Signature=signature”.

Please refer to Appendix 1 (**Python** version sample code) , Appendix 2 (**Golang** version sample code) and Appendix 3 (**Java** version sample code) to build a valid authorization header.

Backend Management

Bioflow leverages backend scheduler to schedule job on any server in computing clusters. Backend management offers 3 APIs: list backends, disable backend, enable backend.

List Backends

List existing backends

GET /v1/backend/list

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below result in JSON format.

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
Backends	[]BackendInfo	Refer to definition of BackendInfo

```
type BackendInfo struct {
        Id string.
        Server string
        Type string
        TaskCount uint64
        Status string
        FailCount uint32
        LastFailAt string
}
```

Disable Backend

Disable a backend

POST /v1/backend/disable/{backendID}

backendID: specify the ID of backend being disabled, no new task would be scheduled to the disabled backend.

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Enable Backend

Enable a disabled backend

POST /v1/backend/enable/{backendID}

backendID: specify the ID of backend being enabled

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Backend Task Notification

Backend Task Notification is the API for scheduler backend, when status of tasks on backend changes, backend would invoke the API to let Bioflow aware of the status change.

```
POST  /v1/tasknotify/{backendID}
```

backendID: Specify notify come from which backend.

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Notification message body differs according different backend.

So far Bioflow supports 2 kinds of backend: Paladin and Kubernetes, the schema of two backend would be introduced separately below.

Encode below struct in JSON format:

Paladin Backend

```
type CallbackParams struct {
        task_id string
        status string
        time int64
}
```

Kubernetes Backend

```
type CallbackParams struct {
        taskId string
        status string
        time string
}
```

Response

HTTP code	Description
200	success
500	internal error

Pipeline Item Management

Pipeline item management manages the components of the pipeline. The API defines how to list/add/delete/info item.

List Pipeline Items

List all pipeline items

GET /v1/item/list

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
PipelineItems	[]PipelineItem	Refer to definition of PipelineItem

```
type PipelineItem struct {
        Name                string
        Cmd                 string
        State               string
        Comments            string
        Cleanup             string
        Owner               string
        ResourceSpec        map[string]interface{}
        GroupPattern        string
        MatchPatter         string
        InputDir            string
        InputDirTag         string
        Filter              string
        OutputFile          string
        OutputFileMap       map[string]string
        OutputDir           string
```

```
    OutputDirTag        string
    ExtensionName       string
    ExtensionMap        map[string]string
    TagPrefix           string
    TagPrefixMap        map[string]string
    InputDirMapTarget   string
    WorkDirMapTarget    string
    Image               string
    Items               []PipelineItem
    Type                string
    BranchVarList       map[string][]string
    BranchVarFiles      map[string]string
    BranchVarTags       map[string]string
    BranchVarMapFile    string
    BranchVarMapTag     string
    InputFile           string
    InputFileTag        string
    WorkDir             string
    ShardGroupSize      int
    FailRetryLimit      int
    FailIgnore          bool
    CPUTuneRatio        float64
    MemTuneRatio        float64
    BranchSelectorFile  string
    BranchSelectorTag   string
    StorageType         string
    ServerType          string
    SortPattern         string
    Privileged          bool
    Env                 map[string]string
    Volumes             map[string]string
    ExecMode            string
    Constraints         map[string]string
    Forward             []string
    Discard             []string
```

```
}
```

PipelineItem is recursive defined because one item can include one or more items inside. More detail about each field please refer to another document "Bioflow Pipeline and Job Specification".

Get Information Of a Pipeline Item

Get the detail information of the item according itemName.

GET /v1/item/info/{pipelineItemId}

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message
PipelineItem	PipelineItem	Refer to definition of PipelineItem

```
type PipelineItem struct {
        Name                    string
        Cmd                     string
        State                   string
        Comments                string
        Cleanup                 string
        Owner                   string
        ResourceSpec            map[string]interface{}
        GroupPattern            string
        MatchPatter             string
        InputDir                string
        InputDirTag             string
        Filter                  string
        OutputFile              string
        OutputFileMap           map[string]string
        OutputDir               string
        OutputDirTag            string
        ExtensionName           string
        ExtensionMap            map[string]string
        TagPrefix               string
        TagPrefixMap            map[string]string
        InputDirMapTarget       string
        WorkDirMapTarget        string
        Image                   string
        Items                   []PipelineItem
        Type                         string
        BranchVarList             map[string][]string
        BranchVarFiles            map[string]string
        BranchVarTags             map[string]string
        BranchVarMapFile       string
        BranchVarMapTag        string
        InputFile                   string
        InputFileTag              string
        WorkDir                     string
        ShardGroupSize          int
        FailRetryLimit          int
        FailIgnore                 bool
        CPUTuneRatio            float64
        MemTuneRatio           float64
        BranchSelectorFile    string
        BranchSelectorTag    string
        StorageType               string
        ServerType                string
        SortPattern               string
        Privileged                 bool
        Env                           map[string]string
        Volumes                     map[string]string
        ExecMode                  string
        Constraints               map[string]string
        Forward                    []string
        Discard                    []string
}
```

PipelineItem is recursive defined because one item can include one or more items inside. More detail about each field please refer to another document "Bioflow Pipeline and Job Specification".

Add Pipeline Item

Add a pipeline Item to Bioflow.

POST /v1/item/add

Parameters

HTTP Header:

Content-Type	application/json
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Items	[]Item	Refer to the define of Item

```
type PipelineItem struct {
        Name                string
        Cmd                 string
        State               string
        Comments            string
        Cleanup             string
        Owner               string
        ResourceSpec          map[string]interface{}
        GroupPattern        string
        MatchPatter         string
        InputDir          string
        InputDirTag       string
        Filter            string
        OutputFile        string
```

```
    OutputFileMap        map[string]string
    OutputDir            string
    OutputDirTag         string
    ExtensionName        string
    ExtensionMap         map[string]string
    TagPrefix            string
    TagPrefixMap         map[string]string
    InputDirMapTarget    string
    WorkDirMapTarget     string
    Image                string
    Items                []PipelineItem
    Type                 string
    BranchVarList        map[string][]string
    BranchVarFiles       map[string]string
    BranchVarTags        map[string]string
    BranchVarMapFile     string
    BranchVarMapTag      string
    InputFile            string
    InputFileTag         string
    WorkDir              string
    ShardGroupSize       int
    FailRetryLimit       int
    FailIgnore           bool
    CPUTuneRatio         float64
    MemTuneRatio         float64
    BranchSelectorFile   string
    BranchSelectorTag    string
    StorageType          string
    ServerType           string
    SortPattern          string
    Privileged           bool
    Env                  map[string]string
    Volumes              map[string]string
    ExecMode             string
    Constraints          map[string]string
```

```
        Forward                 []string
        Discard                 []string
}
```

PipelineItem is recursive defined because one item can include one or more items inside. More detail about each field please refer to another document "Bioflow Pipeline and Job Specification".

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Delete Pipeline Item

POST /v1/item/delete/{pipelineItemId}

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Pipeline Management

Pipeline management offers functions to list/add/delete/clone/update pipeline.

List Pipelines

List all pipelines

GET /v1/pipeline/list

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encoded in below struct in JSON format:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)

Key	Type	Description
Msg	String	Detail error message
Pipelines	[]BioflowPipelineInfo	Refer to definition of the pipeline info

```
type BioflowPipelineInfo struct {
   Name  string
   State string
   Owner string
   Description string
   Type  string
   WorkDir string
   HDFSWorkDir string
   Parent string
   LastVersion string
   Version string
   IgnoreDir string
   InputMap map[string]string
   WorkflowFile string
   ItemCount int

   Items []BioflowPipelineItemInfo
}
```

a Pipeline includes an array of pipelineItem. pipelineItem definition refers to below struct PipelineItem.

```
type PipelineItem struct {
        Name                    string
        Cmd                     string
        State                   string
        Comments                string
        Cleanup                 string
        Owner                   string
        ResourceSpec            map[string]interface{}
        GroupPattern            string
```

```
MatchPatter           string
InputDir              string
InputDirTag           string
Filter                string
OutputFile            string
OutputFileMap         map[string]string
OutputDir             string
OutputDirTag          string
ExtensionName         string
ExtensionMap          map[string]string
TagPrefix             string
TagPrefixMap          map[string]string
InputDirMapTarget     string
WorkDirMapTarget      string
Image                 string
Items                 []PipelineItem
Type                  string
BranchVarList         map[string][]string
BranchVarFiles        map[string]string
BranchVarTags         map[string]string
BranchVarMapFile      string
BranchVarMapTag       string
InputFile             string
InputFileTag          string
WorkDir               string
ShardGroupSize        int
FailRetryLimit        int
FailIgnore            bool
CPUTuneRatio          float64
MemTuneRatio          float64
BranchSelectorFile    string
BranchSelectorTag     string
StorageType           string
ServerType            string
SortPattern           string
```

```
        Privileged              bool
        Env                     map[string]string
        Volumes                 map[string]string
        ExecMode                string
        Constraints             map[string]string
        Forward                 []string
        Discard                 []string
}
```

PipelineItem is recursive defined because one item can include one or more items inside. More detail about each field please refer to another document "Bioflow Pipeline and Job Specification".

Add a Pipeline

Add a pipeline to Bioflow.

POST /v1/pipeline/add

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format, but set different fields for BSL and WDL pipeline.

```
type PipelineJSONData struct {.
       Name  string                                  `json:"Name,omitempty"`
       State int                                     `json:"State,omitempty"`
       UseExistingItem bool
```

```
        InputMap map[string]string                     `json:"InputMap,omitempty"`
        WorkDir string                                 `json:"WorkDir,omitempty"`
        IgnoreDir string                               `json:"IgnoreDir,omitempty"`
        HDFSWorkDir string                             `json:"HDFSWorkDir,omitempty"`
        Description string                             `json:"Description,omitempty"`
        Type  string                                   `json:"Type,omitempty"`
        Itemcount int                                  `json:"Itemcount,omitempty"`
        Items []BioPipelineItemJSONData                `json:"Items,omitempty"`
        Wdl WdlJSONData
}
```

BSL Pipeline

To add a BSL Pipeline should set the following fields:

- **Name**: a string representing the unique id of the pipeline
- **Type**: the const string “BSL”
- **UseExistingItem**: A boolean whether to reuse a pipeline item already added in the system. Default false.
- **WorkDir**: The default work directory of the pipeline
- **Description**: a string to describe the pipeline
- **Items**: an array of pipelineItem. The BioPipelineItemJSONData definition refers to below struct.

```
type BioPipelineItemJSONData struct {

 Name    string                              `json:"Name,omitempty"`
   State  int
   Cmd    string                              `json:"Cmd,omitempty"`
   Comments string                                `json:"Comments,omitempty"`
   ResourceSpec map[string]interface{}              `json:"ResourceSpec"`
   GroupPattern string                             `json:"GroupPattern,omitempty"`
   MatchPattern string                             `json:"MatchPattern,omitempty"`
   SortPattern string                            `json:"SortPattern,omitempty"`
   Filter string                              `json:"Filter,omitempty"`
   Image string                                 `json:"Image,omitempty"`
   Items []BioPipelineItemJSONData                 `json:"Items,omitempty"`
   Type  string                                `json:"Type,omitempty"`
   BranchVarList map[string][]string               `json:"BranchVarList,omitempty"`
   BranchVarFiles map[string]string                `json:"BranchVarFiles,omitempty"`
   BranchVarTags map[string]string                  `json:"BranchVarTags,omitempty"`
   BranchVarMapFile string                       `json:"BranchVarMapFile,omitempty"`
```

```
    BranchVarMapTag string                         `json:"BranchVarMapTag,omitempty"`
    BranchSelectorFile string                      `json:"BranchSelectorFile,omitempty"`
    BranchSelectorTag string                       `json:"BranchSelectorTag,omitempty"`
    Cleanup string                        `json:"Cleanup,omitempty"`
    InputDir string                       `json:"InputDir,omitempty"`
    InputDirTag string                    `json:"InputDirTag,omitempty"`
    OutputFile string                     `json:"OutputFile,omitempty"`
    OutputFileMap map[string]string                 `json:"OutputFileMap,omitempty"`
    OutputDir string                      `json:"OutputDir,omitempty"`
    OutputDirTag string                     `json:"OutputDirTag,omitempty"`
    ExtensionName string                     `json:"ExtensionName,omitempty"`
    ExtensionMap map[string]string                 `json:"ExtensionMap,omitempty"`
    Discard []string                      `json:"Discard,omitempty"`
    Forward []string                      `json:"Forward,omitempty"`
    InputDirMapTarget string                       `json:"InputDirMapTarget,omitempty"`
    WorkDirMapTarget string                        `json:"WorkDirMapTarget,omitempty"`
    InputFile string                      `json:"InputFile,omitempty"`
    InputFileTag string                    `json:"InputFileTag,omitempty"`
    WorkDir string                        `json:"WorkDir,omitempty"`
    ShardGroupSize int                       `json:"ShardGroupSize,omitempty"`
    FailRetryLimit int                    `json:"FailRetryLimit,omitempty"`
    FailIgnore bool
    CPUTuneRatio float64                      `json:"CPUTuneRatio,omitempty"`
    MemTuneRatio float64                       `json:"MemTuneRatio,omitempty"`
    StorageType string                      `json:"StorageType,omitempty"`
    ServerType string                      `json:"ServerType,omitempty"`
    Privileged bool
    Env map[string]string                     `json:"Env,omitempty"`
    Volumes map[string]string                   `json:"Volumes,omitempty"`
    ExecMode string                         `json:"ExecMode,omitempty"`
    TagPrefix string                      `json:"TagPrefix,omitempty"`
    TagPrefixMap map[string]string                 `json:"TagPrefixMap,omitempty"`
    Constraints map[string]string                `json:"Constraints,omitempty"`
    ScheduleDomains string                       `json:"ScheduleDomains,omitempty"`

     IOPattern string                       `json:"IOPattern,omitempty"`

    RWPattern string                        `json:"RWPattern,omitempty"`
    LargeSmallFiles bool                      `json:"LargeSmallFiles,omitempty"`
    WorkingSet string                       `json:"WorkingSet,omitempty"`
    IsolationLevel string                   `json:"IsolationLevel,omitempty"`
    EpheremalLevel string                     `json:"EpheremalLevel,omitempty"`
    EpheremalFilePattern string                  `json:"EpheremalFilePattern,omitempty"`

    EpheremalMap map[string]string                `json:"EpheremalMap,omitempty"`
}
```

Pipeline Item is recursive defined because one item can include one or more items inside. More detail about each field please refer to another document "Bioflow Pipeline and Job Specification".

WDL Pipeline

To add a WDL Pipeline should set the following fields:

- **Name**: a string representing the unique id of the pipeline
- **Type**: the const string "WDL"
- **WorkDir**: The default work directory of the pipeline
- **Description**: a string to describe the pipeline
- **Wdl**: the JSON data structure as follows to describe the WDL pipeline and its source code.

```
type WDLJSONData struct {
    WorkflowFile string                    `json:"WorkflowFile,omitempty"`
    Blob []byte
    StoreDir string                      `json:"StoreDir,omitempty"`

}
```

The Fields should be set as follows:

- ***WorkflowFile***: should be set to the wld file path containing the main workflow relative to the source code directory.
- ***Blob***: The whole source code directory should be compressed to an archive zip file and set "Blob" to the bytes stream. No detailed description here to describe the zip process, but XTao can provide the golang source code of this part to help users understand it.
- ***StoreDir***: not required to be set when adding a pipeline

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Delete a Pipeline

POST /v1/pipeline/delete/{pipelineId}

Delete a pipeline with specific pipelineName

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Info a Pipeline

GET /v1/pipelne/info/{pipelineId}/{version}

Info a pipeline with pipelineName and pipeline version.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message
Pipeline	BioflowPipelineInfo	Detail Bioflow Pipeline Information

HTTP Body:

Encode below struct in JSON format:

```
type PipelineInfo struct {.
        Name                    string
        State                   string
        Owner                   string
        Description             string
        Type                    string
        WorkDir                 string
        HDFSWorkDir           string
        Parent                  string
        LastVersion            string
        Version                string
        IgnoreDir              string
        InputMap                map[string]string
        WorkflowFile          string
        ItemCount              int
        Items                  []PipelineItem
}
```

a Pipeline includes an array of pipelineItem. pipelineItem definition refers to below struct PipelineItem.

```
type PipelineItem struct {
        Name                   string
        Cmd                    string
        State                  string
        Comments               string
        Cleanup                string
        Owner                  string
        ResourceSpec            map[string]interface{}
        GroupPattern           string
        MatchPatter            string
        InputDir             string
        InputDirTag          string
        Filter               string
        OutputFile           string
        OutputFileMap        map[string]string
        OutputDir            string
        OutputDirTag         string
        ExtensionName        string
        ExtensionMap        map[string]string
        TagPrefix            string
        TagPrefixMap        map[string]string
        InputDirMapTarget     string
        WorkDirMapTarget     string
        Image                         string
        Items                         []PipelineItem
        Type                          string
        BranchVarList            map[string][]string
        BranchVarFiles          map[string]string
        BranchVarTags           map[string]string
        BranchVarMapFile     string
        BranchVarMapTag     string
        InputFile                   string
        InputFileTag              string
        WorkDir                     string
        ShardGroupSize        int
        FailRetryLimit          int
        FailIgnore                 bool
        CPUTuneRatio           float64
        MemTuneRatio          float64
        BranchSelectorFile    string
        BranchSelectorTag    string
        StorageType              string
        ServerType               string
        SortPattern              string
        Privileged                bool
        Env                         map[string]string
        Volumes                   map[string]string
        ExecMode                 string
        Constraints             map[string]string
        Forward                   []string
        Discard                   []string
}
```

Clone a Pipeline

POST /v1/pipelne/clone/{srcPipelineId}/{dstPipelineId}

Clone a pipeline from source pipelineName and names new pipeline destination pipelineName.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Update Pipeline

Change a pipeline definition and resubmit it to Bioflow.

POST /v1/pipeline/update

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode in exact the same JSON format with adding pipeline request. Please refer to the section about adding a pipeline.

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Dump a Pipeline

GET /v1/pipelne/dump/{pipelineId}/{version}

Retrieve a pipeline with specific id and version. It will return the data in the same JSON format used when adding the pipeline.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success

HTTP code	Description
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message
PipelineJsonData	PipelineJSONData	The Pipeline data in JSON format

Encode in the JSON format as PipelineJSONData, please check the Add pipeline section for details.

Job Management

List Jobs

List running job or job history.

GET /v1/job/list

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format:

```
{
        ListType string            `json:"All"`
        Name string                `json:"Name"`
        Pipeline string            `json:"Pipeline"`
        After string               `json:"After"`
        Before string              `json:"Before"`
        Finished string            `json:"Finished"`
        Count int                  `json:"Count"`
        SecID string               `json:"Secid" `
        Priority int               `json:"Priority"`
        State string               `json:"State"`
}
```

The Parameter defines job list filter conditions as follows:

- **ListType**: specify the list type, whose value must be one of old|mem|all。"old" means list history jobs, "mem" means list only non-terminated jobs, and "all" means list all the jobs. An empty string means list all jobs.
- **Name**: specify the job name. The empty string "" means ignore the condition.
- **Pipeline**: specify the pipeline name of the job. The empty string "" means ignore the condition.

- **Before/After**: specify the time filter condition which request Bioflow server return only jobs created or finished in the defined valid time. The time condition are all in format like: "2015-07-24 08:00:00" or "2015-07-24". The empty string "" means ignore the condition.
- **Finish**: specify what kind time of job (**After / Before**) condition should be applied to help filter the jobs. "true" means apply the filter on job finish time, while "false" means apply on the job create time.
- **Count**: the maximum count of jobs the list operation is allowed to return. This option is to help control the memory size of result. 0 means don't limit the return job size.
- **SecID**: specify the user name of job owner. The empty string "" means ignore the condition.
- **Priority**: the priority of jobs to list, must take value between 0 ~ 10. A value -1 means ignore the condition.
- **State**: the state of jobs to list. The empty string "" means ignore the condition.

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
Count	Int	Job count in response
Jobs	[]BioflowJobInfo	Array of running jobs
HistJobs	[]BioflowJobInfo	Array of completed and failed jobs.

```
type BioflowJobInfo struct {
        JobId           string
        Name            string
        Pipeline        string
        Created         string
        Finished        string
        State           string
        Description     string
        Owner           string
        Priority        string
        ExecMode        string
        ScheduleDomains string
        Volumes         map[string]string
        EnvVars         map[string]string
        RetryTasks      []string
        HangStages      []BioflowStageInfo
}
```

```
type BioflowStageInfo struct {
        Id                      string
        Name                    string
        State                   string
        Command                  string
        Output                  Map[string]string
        RetryCount              int
        BackendId                string
        SubmitTime              string
        ScheduleTime             string
        FinishTime              string
        TotalDuration           float64
        FailReason              string
        CPU                     float64
        Memory                   float64
        ServerType              string
        GPU                     float64
        GPUMemory               float64
        Disk                    float64
        ExecMode                 string
        HostName                string
        HostIP                  string
        ResourceStats           ResourceUsageInfo
        GlobalResources map[string]uint64
        ScheduleDomains string
}
```

Submit a Job

Submit a job to Bioflow.

POST /v1/job/submit

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format:

```
{
          Name   string                          `json:"Name"`
         Description string                      `json:"Description"`
         Pipeline string                         `json:"Pipeline"`
         WorkDir string                          `json:"WorkDir"`
         LogDir string                            `json:"LogDir"`
        InputDataSet DataSetJSONData             `json:"InputDataSet"`
        SMId string                              `json:"SMId"`
        Priority int
        DisableOOMCheck bool
        ExecMode string
        HDFSWorkDir string                       `json:"HDFSWorkDir"`
        StageQuota *int                          `json:"StageQuota"`
        Constraints map[string]string            `json:"Constraints"`
        ScheduleDomains string                   `json:"ScheduleDomains"`
        Volumes       map[string]string          `json:"Volumes"`
        EnvVars      map[string]string           `json:"EnvVars"`

}
```

The fields above should be set as follows:

- **Name**: the name of the job, needn't be unique, allowed empty string.
- **Description**: description of the job, allowed empty string.
- **Pipeline**: the pipeline of the job to execute on, must be the name of a existing valid pipeline in the system. Pipeline is case insensitive.

- **WorkDir**: the work directory of the job. Bioflow will automatically create a work directory for the job if it is set empty string here.
- **LogDir**: the directory to store the logs of the job. Bioflow will automatically create one under job's work directory if not set here.
- **InputDataSet**: the input parameters of the job, BSL and WDL job should be set in different way. Please refer the following section.
- **SMId**: sample id of the BSL job. WDL job just ignore it and let it empty.
- **Priority**: the priority of the job, default value is 0, which is the lowest.
- **DisableOOMCheck**: a flag set to disable OOM killer for the job. Recommend to let it be default value false, unless know the risk to enable it.
- **ExecMode**: the execution mode of the job, can be "process", "docker" or "singularity". The empty value will let bioflow to choose the default mode "docker". This setting will override the setting in the pipeline.
- **HDFSWorkDir**: the work directory of the job which will be executed on HDFS. Suggest keep it empty value to let Bioflow automatically determinate the value.
- **StageQuota**: the maximum concurrent stages of the job can be scheduled to run at the sometime. Don't set it (let it empty in JSON) or set it to a largest value will make Bioflow determine it. Bioflow will determine the final value according to policy set by administrator even it is set here.
- **Constraints**: the nodes in Achelous cluster can be tagged with key/value pair. Job can be submit with a series of key/value pair to help select nodes to execute the job. **Suggest don't set it** unless knowing the risk.
- **ScheduleDomains**: the name of the schedule domain or queue to execute the job.
- **Volumes**: the volumes map of the job, which will be set for every docker/ singularity container of the job. ***Key*** is the container path, while ***Value*** is the host path.
- **EnvVars**: the environment variables which will be set for every task of the job.

The pipeline can be wrote in BSL or WDL language, which will be feed the input dataset in the job JSON in a different way.

```
DataSetJSONData {
        Files []string                                  `json:"Files" `
        FilesInOrder[][]string                          `json:"FilesInOrder"`
        InputDir string                                 `json:"InputDir"`
        Vol string                                      `json:"Vol"`
        InputMap map[string]string                      `json:"InputMap"`
        WorkflowInput map[string]interface{}            `json:"WorkflowInput"`
        RestorePath string                              `json:"RestorePath"`
        RestoreFilter []string                        `json:"RestoreFilter"`
        LogPath string                                `json:"LogPath"`

}
```

The above JSON structure should be set as follows:

- **Files**: the input file list of job, **only used** when job's input data is stored in the OSS, which need to be fetched automatically to local cache file system store before read/write it.
- **FilesInOrder**: some of files specified in **Files** may need to be download in order. The order should be specified here.
- **InputDir**: For *BSL* jobs to specify input data directory
- **Vol**: For *BSL* jobs to specify the volume of input data
- **InputMap**: For *BSL* jobs to specify the input parameters of the pipeline
- **WorkflowInput**: For *WDL* jobs to specify the input or parameters of the workflow
- **RestorePath**: For jobs which need store output result to OSS storage, specify the store bucket here.
- **RestoreFilter**: work with **RestorePath** to specify pattern to match the output files which need to be restored to the OSS storage.
- **LogPath**: The bucket path to restore the logs of the job

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success

HTTP code	Description
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
JobId	String	job UUID

Cancel a Job

POST /v1/job/cancel/{jobId}

Cancel a submitted job with specific jobId.

jobId is a UUID which is relatively long. The API accepts "part of UUID", Bioflow server would fuzzy match the inputed jobID as long as the match result is unique.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Get Status of a Job

GET /v1/job/status/{jobId}

Get the status of a job with specific jobId. The status information includes detail status information of each stages of the job.

jobId is a UUID which is relatively long. The API accepts “part of UUID”, Bioflow server would fuzzy match the inputed jobID as long as the match result is unique.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
JobStatus	BioflowJobStatus	Detail Job Status Information

Encode below struct in JSON format:

```
type BioflowJobStatus struct {
        JobId                   string
        Name                    string
        Pipeline                string
        Owner                   string
        WorkDir                 string
        HDFSWorkDir             string
        Created                 string
        Finished                string
        State                   string
        PausedState             string
        RunCount                int
        Priority                int
        RetryLimit              int
        StageCount              int
        StageQuota              int
        ExecMode                string
        ScheduleDomains         string
        Volumes                 map[string]string
        EnvVars                 map[string]string
        RetryTasks              []string
        FailReason              []string
        GraphCompleted          bool
        PipelineBuildPos        int
        DoneStages              []BioflowStageInfo
        PendingStages           []BioflowStageInfo
        WaitingStages           []BioflowStageInfo
        ForbiddenStages         []BioflowStageInfo
        JobOutput               map[string]string
        Constraints             map[string]string
}
```

- State is current state of the job. Valid State are:
  - "CREATED"
  - "RUNNING"

- "FAIL"
- "FINISHED"
- "LOST"
- "RECOVERY"
- "PAUSED"
- "CANCELED"
- "PSUDONE"

- PausedState is the job state before being paused. Valid PausedState are:
  - "CREATED"
  - "RUNNING"
- DoneStages are the completed or failed stages.
- PendingStages are the stages submitted to backend but haven't been completed or failed.
- WaitingStages are the stages that are not ready to schedule to backend since dependencies haven't been satisfied.
- ForbiddenStages are the stages that cannot continue run because some stages fail.
- GraphCompleted is a flag to hint the scheduler execution graph is partially build. The job is consist of several stages, and there are dependencies among stages, scheduler schedules the stages of job according a DAG. However, the graph might not be built completely before execution because ShardFiles item is in the middle of the pipeline definition, which means the graph can be partially build before being submitted to the scheduler. Graph builder would resume graph building only when input files of the ShardFiles item are generated.
- PipelineBuildPos, the attribute takes effect only when GraphCompleted set to False. GraphCompleted being set to False implies the graph builder suspends graph building process for the job until premise is met. PipelineBuildPos is to record last item of pipeline where graph builder builds graph to. PipelineBuildPos is actually the item sequence number in the pipeline.

```
type BioflowStageInfo struct {
        Id                  string
        Name                string
        Image               string
        State               string
        Command             string
        Output              Map[string]string
        RetryCount             int
        BackendId               string
        TaskId                 string
        SubmitTime             string
        ScheduleTime            string
        FinishTime              string
        RequestCPUTime      float64
        TotalDuration         float64
        FailReason             string
        CPU                    float64
        Memory                  float64
        ServerType             string
        GPU                    float64
        GPUMemory              float64
        Disk                   float64
        ExecMode                string
        HostName               string
        HostIP                 string
        ResourceStats          ResourceUsageInfo
        GlobalResources      map[string]uint64
        ScheduleDomains      string
}
```

● State, Valid stage states are:

- "STAGE_INITIAL"
- "STAGE_FAIL"
- "STAGE_DONE"
- "STAGE RUNNING"
- "STAGE_LOST"
- "STAGE_SUBMITTED"
- "STAGE_QUEUED"

The ResourceUsageInfo struct is defined as follows:

```
type ResourceUsageInfo struct {

    AvgCPURatio                 float64
    MaxCPURatio                 float64
    LowCPURatio                 float64
    TotalCPUTime                float64
    Threads                     int
    MaxMem                      float64
    MaxIOMem                    float64
    AvgMem                      float64
    AvgIOMem                    float64
    MaxSwap                     float64
    AvgSwap                     float64
    TotalRead                   float64
    TotalWrite                  float64
    TotalSysCR                  float64
    TotalSysCW                  float64
    OOM                         bool
    MaxProfilingCost            float64
    GpuUtilization              map[string]float64
    GpuMemoryUtilization        map[string]float64
    GpuMaxMemoryUsage           map[string]float64
    Periods                     []PeriodResourceUsageInfo

}
```

```
type PeriodResourceUsageInfo struct {

      AvgSysRatio               float64
      AvgUserRatio              float64
      TotalCpuTime              float64
      MaxMem                    float64
      AvgMem                    float64
      MaxCache                  float64
      AvgCache                  float64
      MaxSwap                   float64
      AvgSwap                   float64
      TotalRead                 float64
      TotalWrite                float64
      TotalSysCR                float64
      TotalSysCW                float64
}
```

Pause a Running Job

POST /v1/job/pause/{jobId}

Pause a running or created job with specific jobId.

jobId is a UUID which is relatively long. The API accepts "part of UUID", Bioflow server would fuzzy match the inputed jobID as long as the match result is unique.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Resume a Paused Job

POST /v1/job/resume/{jobId}

Resume a paused job with specific jobId.

jobId is a UUID which is relatively long. The API accepts "part of UUID", Bioflow server would fuzzy match the inputed jobID as long as the match result is unique.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success

HTTP code	Description
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Recover a Failed Job

POST /v1/job/recover

Recover one or more failed jobs satisfy the specified conditions.

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format:

```
{

     After string            `json:"After"`

     Before string           `json:"Before"`
     Pipeline string         `json:"Pipeline"`
     Count int               `json:"Count"`
    JobId string             `json:"Jobid"`
    JobOpt string            `json:"Jobopt"`
    RetryTasks []string      `json:"RetryTasks"`

    ForwardEpoch bool        `json:"ForwardEpoch"`


}
```

The fields should be set as follows:

- **Before/After**: specify the time filter condition which request Bioflow server recover only jobs created in the defined valid time. The time condition are all in format like: "2015-07-24 08:00:00" or "2015-07-24". The empty string "" means ignore the condition.
- **Pipeline**: recover all the failed jobs of the specified pipeline. The empty string "" means ignore the condition.
- **Count**: specify the maximum number of jobs can recovered in the operation. A 0 value means don't limit the number.
- **JobId**: recover the job with the specified ID.
- **JobOpt**: a string specifies the option to recover the job. It can be "default" or "skippendingstages". The "default" option means recovering job from last failed stages. The "skippendingstages" option means ignore the current pending stages and recover from the done stages when the job failed. User should always recover the job with "default" option. If the recover failed, the user can try recover the job with "skippendingstages" option as the last resort.
- **RetryTasks**: a list of tasks to retry execute when recovering the job, even the tasks specified already finished.
- **ForwardEpoch**: a boolean value to indicate whether ignore all the previous task re-execution state. True means ignore. Default is false. It is only used tougher with **RetryTasks**.

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Re-queue a Queued Job

POST /v1/job/requeue/{jobId}/{stageId}

Cancel a submitted job with specific jobId if the job is in queued state for a long time, then re-queue the job stage to backend for execution.

jobId is a UUID which is relatively long. stageId is bioflow own define special ID which can be found by #biocli job status <jobID>. The API accepts "part of UUID", Bioflow server would fuzzy match the inputed jobID as long as the match result is unique.

Parameters

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format:

```
type ResourceSpecJSONData struct {
        Cpu                     float64     /* Update Cpu of the stage */
        Memory                  float64     /* Update Memory of  the stage */
        Disk            float64     /* Update Disk of  the stage*/
}
```

Response

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Get Job Resource Usage

GET /v1/job/resource/{jobId}/{taskId}

Get the resource usage info of a task of a job. The jobId and taskId should be specified in the URL, while taskId should point to a valid task. Pattern match is not supported for taskId. The taskId can be obtained through job status API.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success

HTTP code	Description
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
Usage	map[string]ResourceUsageInfo	The resource usage statistics of the task of the job

The resource usage statistics is Encoded in JSON as follows:

```
type ResourceUsageInfo struct {

    AvgCPURatio             float64
    MaxCPURatio             float64
    LowCPURatio             float64
    TotalCPUTime            float64
    Threads                 int
    MaxMem                  float64
    MaxIOMem                float64
    AvgMem                  float64
    AvgIOMem                float64
    MaxSwap                 float64
    AvgSwap                 float64
    TotalRead               float64
    TotalWrite              float64
    TotalSysCR              float64
    TotalSysCW              float64
    OOM                     bool
    MaxProfilingCost        float64
    GpuUtilization          map[string]float64
    GpuMemoryUtilization    map[string]float64
    GpuMaxMemoryUsage       map[string]float64
    Periods                 []PeriodResourceUsageInfo
}
```

The data above is the cumulative statistics by summing over the data of all the periods. If user care about the resource statistics of each period, please check the data in ***Period*** field. It is an array consist of the following JSON format data. Each period is about 10 minutes long. The array is organized in time order.

```
type PeriodResourceUsageInfo struct {

        AvgSysRatio              float64
        AvgUserRatio             float64
        TotalCpuTime             float64
        MaxMem                   float64
        AvgMem                   float64
        MaxCache                 float64
        AvgCache                 float64
        MaxSwap                  float64
        AvgSwap                  float64
        TotalRead                float64
        TotalWrite               float64
        TotalSysCR               float64
        TotalSysCW               float64
}
```

Get Job Monitor

GET /v1/job/show/monitor

Gets the tasks from the failed state of all jobs running in the system as well as the tasks from the running state.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
Monitor	map[string][]MonitorUsageInfo	Map of Monitor, key is job id.

Encode below struct in JSON format:

```
type JobMonitorInfo struct {
        JobId                   string
        PipelineName            string
        TaskName                string
        TaskStatus              string
        TaskOwner               string
        OwnerGroup              string
        Submited                string
        Scheduled               string
        Health                          string
        MonitorResourceUsage            *MonitorResourceInfo
}
type JobMonitorInfo struct {
        Threads                         int
        RequestCPUTime                  float64
        TotalCpuTime                    float64
        AvgCPURatio                     float64
        MaxCPURatio                     float64
        AvgMem                          float64
        MaxMem                          float64
        TotalRead                       float64
        TotalWrite                      float64
        MaxSwap                         float64
        GpuUtilization                  map[string]float64
        GpuMemoryUtilization            map[string]float64
        GpuMaxMemoryUsage               map[string]float64
}
```

Dump Job Monitor

GET /v1/job/dump/monitor

Get all the tasks of all jobs in the operation of the system, including failed, finished, running, forbudden and waiting status.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
Monitor	map[string][]MonitorUsageInfo	Map of Monitor, key is job id.

Encode below struct in JSON format:

```
type JobMonitorInfo struct {
        JobId                   string
        PipelineName            string
        TaskName                string
        TaskStatus              string
        TaskOwner               string
        OwnerGroup              string
        Submited                string
        Scheduled               string
        Health                          string
        MonitorResourceUsage            *MonitorResourceInfo
}
type JobMonitorInfo struct {
        Threads                         int
        RequestCPUTime                  float64
        TotalCpuTime                    float64
        AvgCPURatio                     float64
        MaxCPURatio                     float64
        AvgMem                          float64
        MaxMem                          float64
        TotalRead                       float64
        TotalWrite                      float64
        MaxSwap                         float64
        GpuUtilization                  map[string]float64
        GpuMemoryUtilization            map[string]float64
        GpuMaxMemoryUsage               map[string]float64
}
```

Get Job Error Log

GET /v1/job/logs/{jobId}/{stageName}/{stageId}/{taskId}

Get logs for a failed job with specific jobId, specific stageId and specific taskId.

- **stageName/stageId**: retrieve the log of stage identified by the specified name or id. If set "*",the API retrieve the logs of all stages of the job.
- **taskId**: retrieve the log of specific task. If the taskId is "*", try to retrieve the logs of all tasks, which means that the taskId will not be checked.

If stageId, stageName or taskId is not "*", the API will get the log of stages satisfying the conditions simultaneously.

When a stage fails, the stage could be retried under certain circumstance, so there could be more than one stage instances associate to one stage. The API is to retrieve logs for all stage instances of failed stages.

Parameters

N/A

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
JobLogs	map[string]StageLog	Map of StageLog, key is stageName.

```
type StageLog struct {
        StageName           string
        Logs                map[string]string
}
```

The key of Logs is stage instanceID which is UUID which is generated by the backend.

Cleanup job history

POST /v1/job/cleanup

This API is called by user to delete the history of job stored in the database. **Each caller user only cleanup his/her own job, while the root user can cleanup all users history.**

Bioflow will keep all the history of job execution in the database. But each database has a capacity limit, so suggest user recycle the history of jobs which may not be accessed in the future. The recycled job history will not be shown in the result of job list API.

The cleanup API has the same input parameters with job list API, which help user to select jobs with conditions.

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8
X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP Body:

Encode below struct in JSON format:

```
{
        ListType string            `json:"All"`
        Name string                `json:"Name"`
        Pipeline string            `json:"Pipeline"`
        After string               `json:"After"`
        Before string              `json:"Before"`
        Finished string            `json:"Finished"`
        Count int                  `json:"Count"`
        SecID string               `json:"Secid" `
        Priority int               `json:"Priority"`
        State string               `json:"State"`
}
```

The Parameter defines job cleanup filter conditions as follows:

- **ListType**: Not applied here. API only cleanup history jobs.
- **Name**: specify the job name. The empty string “” means ignore the condition.
- **Pipeline**: specify the pipeline name of the job. The empty string “” means ignore the condition.
- **Before/After**: specify the time filter condition which request Bioflow server return only jobs created or finished in the defined valid time. The time condition are all in format like: “2015-07-24 08:00:00” or “2015-07-24”. The empty string “” means ignore the condition.
- **Finish**: specify what kind time of job (**After / Before**) condition should be applied to help filter the jobs. “true” means apply the filter on job finish time, while “false” means apply on the job create time.
- **Count**: the maximum count of jobs the list operation is allowed to return. This option is to help control the memory size of result. 0 means don’t limit the return job size.
- **SecID**: specify the user name of job owner. The empty string “” means ignore the condition.
- **Priority**: the priority of jobs to cleanup, must take value between 0 ~ 10. A value -1 means ignore the condition.

- **State**: the state of jobs to cleanup. The empty string “” means ignore the condition.

Response

HTTP Header:

X-XTAO-Account	Refer to Authorization Header
Authorization	Refer to Authorization Header

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message
CleanupCount	Integer	The count of jobs cleanup

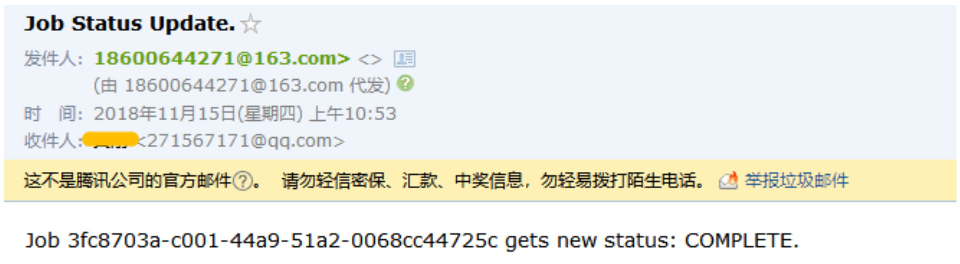
E-mail notify for job

When status of a job get changed, we will send a e-mail to user for status update. User can use command ‘biocli’ to register a e-mail account to receive the notify.

Format of e-mail content

Job {job ID} gets new status: {status}.

Example:



Job Status Update. ☆
发件人: 18600644271@163.com> <>
(由 18600644271@163.com 代发)
时 间: 2018年11月15日(星期四) 上午10:53
收件人:[redacted]<271567171@qq.com>
这不是腾讯公司的官方邮件ⓘ。 请勿轻信密保、汇款、中奖信息，勿轻易拨打陌生电话。 举报垃圾邮件

Job 3fc8703a-c001-44a9-51a2-0068cc44725c gets new status: COMPLETE.

Rest API notify for job and stage

When status of a job or stage get changed, we will post a status information to a rest API for status update. User can use command ‘biocli’ to register a rest API URL to receive the notify.

Format of HTTP body

Json format of struct ‘APINoticeInfo’:

```
type APINoticeInfo struct {
        Type                    string
        JobInfo                 *JobNoticeInfo
        StageInfo               *StageNoticeInfo
}
type JobNoticeInfo struct {
        JobId                   string
        Name                    string
        Pipeline                string
        Owner                   string
        WorkDir                 string
        HDFSWorkDir             string
        Created                 string
        Finished                string
        State                       string
        PausedState                 string
        RunCount                    int
        Priority                    int
        RetryLimit                  int
        StageCount                  int
        ExecMode                    string
        FailReason                  []string
        GraphCompleted              bool
        PipelineBuildPos            int
}
```

```
type StageNoticeInfo struct {
   JobId                          string
   JobName                        string
   Id                             string
   Name                           string
   State                          string
   Command                        string
   Output                         map[string]string
   RetryCount                     int
   BackendId                      string
   TaskId                         string
   SubmitTime                     string
   ScheduleTime                   string
   FinishTime                     string
   TotalDuration                  float64
   FailReason                     string
   CPU                            float64
   Memory                         float64
   ServerType                     string
   ExecMode                       string
```

```
    HostName                 string
    HostIP                   string
    ResourceStats            ResourceUsageInfo
}
```

User Management

User Management offers functions to load user, list all users, show user info, list users' resource stats, reset users' resource accounting, and Configure the user info six API.

List All users

list all exsiting bioflow users.

```
GET  /v1/user/list
```

.

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message
UserList	[]BioflowUserInfo	Refer to definition of BioflowUserInfo

```
type BioflowUserInfo struct {
   Username                        string
   Uid                             string
   Gid                             string
   Groupname                       string
   JobNum                          int64
   TaskNum                         int64
   PauseJobNum                     int64
   CompletedJobNum                 int64
```

```
    LostTaskNum                 int64
    CanceledJobNum                int64
    FailJobNum                   int64
    RunningJobNum                int64
    Credit                       int64
    JobQuota                       int64
    TaskQuota                    int64
    CpuQuota                     int64
    MemQuota                     int64
    GpuQuota                     int64
    GpuMemQuota                  int64
    UsedCpu                      float64
    UsedMem                      float64
    UsedGpu                       int64
    UsedGpuMem float64
    Mail                         string
    Groups                         map[string]bool
}
```

Show the user info

GET /v1/user/info/{userId}

Get user info with specific userId.

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message
UserInfo	BioflowUserInfo	Refer to definition of BioflowUserInfo

```
type BioflowUserInfo struct {
    Username                  string
    Uid                       string
    Gid                       string
    Groupname                 string
    JobNum                    int64
    TaskNum                   int64
    PauseJobNum               int64
    CompletedJobNum           int64
    LostTaskNum               int64
    CanceledJobNum            int64
    FailJobNum                int64
    RunningJobNum             int64
    Credit                    int64
    JobQuota                  int64
```

```
  TaskQuota                  int64

  CpuQuota                   int64
  MemQuota                   int64
  GpuQuota                   int64
  GpuMemQuota                int64
  UsedCpu                    float64
  UsedMem                    float64
  UsedGpu                    int64

  UsedGpuMem float64

  Mail                       string

  Groups                     map[string]bool
}
```

List users' resource stats

GET /v1/user/listrscstats

List all users resource stats include JobCount,StageCount,CPUMinutes,MemMinutes.

Parameters

N/A

Response

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP code	Description
202	success
500	internal error

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message
StatsList	[]UserRscStats	Refer to definition of UserRscStats

```
type UserRscStats struct {
   Id                     string
   JobCount               uint64
   StageCount             uint64
   CpuMinutes             float64
   MemMinutes             float64
   GpuMinutes             float64
   GpuMemMinutes          float64
}
```

Reset users' resource accounting

POST /v1/user/resetrscstats

Cleanup all users resource stats.

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP Body:

Encode below struct in JSON format:

```
{
        UserId              string       /* cleanup resource with defined user name */
        After               string       /* cleanup resource defined time */
        Before              string       /* cleanup resource defined time */
}
```

Parameter defines cleanup resource filter conditions. "Before" and "After" are time filter condition which request job scheduler time in the defined valid time. The time condition are all in format like:

[2017-06-01 12:00:00, -]

Response

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: ("OK", "FAIL")
Msg	String	Detail error message

Configure the user info

POST /v1/user/updateconfig

Configure the user info, include credit, jobQuota,taskQuota and mail.

The valid credit value range is integers: 0 ~ 11. 0 means credit value is disabled and not used. other credit value n means the user job's highest priority is n -1. For example: user's credit value is set 4. If user submit a job with priority 1, then the job priority is 1. if user submit a job with priority 6, the priority is set to 3 (4 - 1). So the credit value restrict user job priorities' upper limit.

Parameters

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP Body:

Encode below struct in JSON format:

```
{
        ID                    string         /* update with defined user name */
        Credit                int64          /* The credit vaule of user */
        JobQuota                     int64         /* The job quota of user */

        TaskQuota             int64          /* The task quota of user */

        CpuQuota              int64          /* maximun allowed cpus at a time*/
        MemQuota              int64          /* maximun allowed mem (MB) at a time*/
        GpuQuota              int64          /* maximun allowed gpus at a time*/
        GpuMemQuota           int64       /* maximun allowed GPU Mem (MB) at a time*/


        Mail                  string         /* The mail address that receive notice */
}
```

Response

HTTP Header:

Content-Type	application/json; charset=UTF8

HTTP Body:

Encode below struct in JSON format:

Key	Type	Description
Status	String	Enum: (“OK”, “FAIL”)
Msg	String	Detail error message

Appendix 1. Python Version Sample Code

```
from Crypto.Cipher import AES
import base64
import binascii
import json
import requests
import hmac
import hashlib
from urllib.parse import urlparse
from pprint import pprint

A_KEY = "96b8a894eb"
S_KEY = "24d75acd04"
D_KEY = "%16s" % S_KEY

XTAO_REST_API_SCHEME = "http"
XTAO_REST_API_HOST = "192.168.247.1"
XTAO_REST_API_PORT = 1028

XTAO_REST_API_USER_ACCOUNT = {"user": "eagle", "group": "eagle", "uid": "500",
"gid": "1000", "umask": "0002"}


BLOCK_SIZE = 16
PADDING = '\0'
padding = lambda s: s+(16 - len(s)%16)*PADDING

def encrypt_by_dkey(data):
        cipher = AES.new(D_KEY, AES.MODE_CFB, D_KEY, segment_size=128)
        encryptd_str = cipher.encrypt(padding(data))
        result = binascii.hexlify(encryptd_str)
        result = result.decode('utf-8')[:2*len(data)]
        return result

def signature_request_str(request_str):
        cipher = hmac.new(S_KEY.encode('utf-8'), request_str.encode('utf-8'),
digestmod=hashlib.sha256)
        signature_str = base64.b64encode(cipher.digest())
        return signature_str.decode('utf-8')

def get_header_str_by_http_headers(http_headers):
        return http_headers['X-XTAO-Account']
        # return header_str


def get_str_by_request(http_request):
        url = urlparse(http_request.url)
        request_str = "{}\n{}\n".format(
                "\n".join([http_request.method, url.netloc, url.path]),
                get_header_str_by_http_headers(http_request.headers)
        )
        return request_str


def get_response(method, path, data=None):
        XTAO_REST_API_AHEADER =
encrypt_by_dkey(json.dumps(XTAO_REST_API_USER_ACCOUNT))
        XTAO_REST_API_PATH = path
        XTAO_REST_API_METHOD = method

        XTAO_REST_API_URL = "{scheme}://{host}:{port}{path}".format(
                scheme = XTAO_REST_API_SCHEME,
                host   = XTAO_REST_API_HOST,
                port   = XTAO_REST_API_PORT,
                path   = XTAO_REST_API_PATH
        )
        headers = {}
        headers["X-XTAO-Account"] = XTAO_REST_API_AHEADER
        # headers["Content-Type"] = "application/json;charset=UTF-8"
        request = requests.Request(method=XTAO_REST_API_METHOD,
url=XTAO_REST_API_URL, headers=headers, data=json.dumps(data))
        request_str = get_str_by_request(request)

        signature = signature_request_str(request_str)
        request.headers['Authorization'] = "APIKey={},Signature={}".format(A_KEY,
signature)

        prepared_request = request.prepare()
        session = requests.Session()
        response = session.send(prepared_request)
        return response

def add_pipeline():
        path = "/v1/pipeline/add"
        method = "POST"
        data = {
      "Name": "SAMTOOLS_PIPELINE",
      "Description": "test bioflow pipeline",
      "WorkDir":"xtvol@xtao:eagle/jobs",
      "Items": [
            {
                  "Name": "samtools-view",
                  "Image": "samtools:0.1.19",
                  "Cmd": "samtools view -bS $file.sam -o $output.bam"
            }
      ]

        }
        res = get_response(method, path, data)
        # pprint(res.status_code)
        pprint(json.loads(res.content))
```

```
if __name__ == '__main__':
        add_pipeline()
```

Appendix 2. Golang Version Sample Code

```
package client

import (
    "fmt"
    "bytes"
    "strings"
    "net/http"
    "crypto/hmac"
    "crypto/sha256"
    "encoding/base64"
    "encoding/json"
)

const (
        AccountHeader = "X-XTAO-Account"
)
const (
        // common parameters
        authorizationHeader = "Authorization"
        apiKeyParam         = "APIKey"
        signatureParam      = "Signature"
        accountHeader       = AccountHeader
        // parsing bits
        empty   = ""
        comma   = ","
        space   = " "
        eqSign  = "="
        newline = "\n"
)

func SignString(str string, secret string) string {
        hash := hmac.New(sha256.New, []byte(secret))
        hash.Write([]byte(str))
        return base64.StdEncoding.EncodeToString(hash.Sum(nil))
}

func StringToSign(req *http.Request, options *Options) (string, error) {
        var buffer bytes.Buffer

        // Standard
        buffer.WriteString(req.Method)
        buffer.WriteString(newline)
        buffer.WriteString(req.Host)
        buffer.WriteString(newline)
        buffer.WriteString(req.URL.RequestURI())
        buffer.WriteString(newline)

        // Headers
        sort.Strings(options.SignedHeaders)
        for _, header := range options.SignedHeaders {
                val := req.Header.Get(header)
                if val == empty {
                        return empty, HeaderMissingError{header}
                }
                buffer.WriteString(val)
                buffer.WriteString(newline)
        }

        return buffer.String(), nil
}

func SignClientReqHeader(req *http.Request, account *UserAccountInfo,
   string aKey, string sKey) error {
   var user, group, uid, gid, umask string

   signedHeaders := []string{}

   if _, exists := req.Header["Content-Type"]; exists {
      signedHeaders = append(signedHeaders, "Content-Type")
   }
   accHeader := make(map[string]interface{})

   user = UID_GLOBAL
   if account.Username != "" {
      user = account.Username
   }

   accHeader["user"] = user

   if account.Groupname != "" {
      group = account.Groupname
      accHeader["group"] = group
   }

   if account.Uid != "" {
      uid = account.Uid
      accHeader["uid"] = uid
   }

   if account.Gid != "" {
      gid = account.Gid
      accHeader["gid"] = gid
   }

   if account.Umask != "" {
      umask = account.Umask
      accHeader["umask"] = umask
   }

   js, err := json.Marshal(accHeader)
```

```
    if err != nil {
        return err
    }

    // Encrypt the account information
    dKey := sKey
    if len(dKey) < 16 {
        dKey = fmt.Sprintf("%16s", sKey)
    }

    aesEnc := NewAesEncrypt(dKey)
    aHeader, err := aesEnc.Encrypt(string(js))
    if err != nil {
        return err
    }

    signedHeaders = append(signedHeaders, AccountHeader)
    req.Header.Add(AccountHeader, aHeader)

    options := Options {
        SignedHeaders: signedHeaders,
    }

    str, err := StringToSign(req, &options)
    if err != nil {
        return err
    }

    signature := SignString(str, sKey)

    authHeader := fmt.Sprintf("APIKey=%s,Signature=%s", aKey, signature)
    req.Header.Add("Authorization", authHeader)

    return nil
}
```

Appendix 3. Java Version Sample Code

```
package com.xtao.bioflow;

import java.io.IOException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

import org.apache.http.client.ClientProtocolException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.xtao.bioflow.constant.BiowflowInputMap;
import com.xtao.bioflow.constant.JobState;
import com.xtao.bioflow.constant.StageState;
import com.xtao.bioflow.dto.ListJobParam;
import com.xtao.bioflow.exception.BioflowException;
import com.xtao.bioflow.util.BioflowSignatureUtils;
import com.xtao.bioflow.util.RestInvokerUtils;

public class BioflowManager {
	private static Logger logger = LoggerFactory.getLogger(BioflowManager.class);
	private String domain;
	private String gid;
	private String group;
	private String uid;
	private String umask;
	private String user;
	private String apiKey;
	private String securityKey;
	private String dKey;

	String accHeader = "{\"gid\":\"1000\",\"group\":\"1000\",\"uid\":\"500\",\"umask\":\"2\",
\"user\":\"taihe\"}";

	private final static String PROTOCOL = "http://";

	private final static String URI_LIST_JOB = "/v1/job/list";

	private final static String FIELD_STATUS = "Status";
	private final static String FIELD_MSG = "Msg";
	private final static String FIELD_COUNT = "Count";
	private final static String FIELD_JOBS = "Jobs";
	private final static String FIELD_HIST_JOBS = "HistJobs";
	private final static String FIELD_JOB_ID = "JobId";
	private final static String FIELD_JOB_STATUS = "JobStatus";
	private final static String FIELD_JOB_LOGS = "JobLogs";

	private final static String FIELD_GID = "gid";
	private final static String FIELD_GROUP = "group";
	private final static String FIELD_UID = "uid";
	private final static String FIELD_UMASK = "umask";
	private final static String FIELD_USER = "user";

	private final static String HEADER_AUTHORIZATION = "Authorization";
	private final static String HEADER_X_XTAO_ACCOUNT = "X-Xtao-Account";
	private final static String HEADER_CONTENT_TYPE = "Content-Type";

	private final static String CONTENT_TYPE_JSON = "application/json";

	private final static String LINE_DMT = "\n";

	public BioflowManager(String domain, String user, String apiKey, String securityKey) {
		this(domain, user, apiKey, securityKey, "1001", "1001", "1001", "2");
	}

	public BioflowManager(String domain, String user, String apiKey, String securityKey, String
group, String gid,
			String uid, String umask) {
		this.domain = domain;
		this.user = user;
		this.apiKey = apiKey;
		this.securityKey = securityKey;
		this.group = group;
		this.gid = gid;
		this.uid = uid;
		this.umask = umask;

		this.dKey = this.securityKey;
		if (this.dKey.length() < 16) {
			this.dKey = String.format("%16s", this.securityKey);
		}
	}

	private String getAccHeader() {
		JSONObject jsonObject = new JSONObject();
		jsonObject.put(FIELD_GID, this.gid);
		jsonObject.put(FIELD_GROUP, this.group);
		jsonObject.put(FIELD_UID, this.uid);
		jsonObject.put(FIELD_UMASK, this.umask);
		jsonObject.put(FIELD_USER, this.user);
		return jsonObject.toJSONString();
	}

	private String getEncryptAccHeader() throws InvalidKeyException,
InvalidAlgorithmParameterException,
			NoSuchPaddingException, NoSuchAlgorithmException,
IllegalBlockSizeException, BadPaddingException {
		String accHeader = getAccHeader();

		return BioflowSignatureUtils.toHex(BioflowSignatureUtils.encrypt(accHeader,
this.dKey));
	}

	private String getAuthorizationHeader(String method, String uri, Map<String, String>
headers)
			throws InvalidKeyException, NoSuchAlgorithmException {
		StringBuilder sb = new StringBuilder();
		sb.append(method);
		sb.append(LINE_DMT);
		sb.append(this.domain);
		sb.append(LINE_DMT);
		sb.append(uri);
		sb.append(LINE_DMT);
		Set<String> keySet = headers.keySet();
		Object[] keyArray = keySet.toArray();
```

```
				Arrays.sort(keyArray);
				for (Object key : keyArray) {
					String value = headers.get(key);
					sb.append(value);
					sb.append(LINE_DMT);
				}

				return BioflowSignatureUtils.hmacSha256(this.securityKey.getBytes(), 
sb.toString().getBytes());
		}

		private JSONObject responseParse(String responseStr) throws BioflowException {
//				System.out.println("=== responseStr : " + responseStr);
				JSONObject jsonObject = JSON.parseObject(responseStr);
				String status = jsonObject.getString(FIELD_STATUS);
				if (status == null || !status.equals("OK")) {
					throw new BioflowException(jsonObject.getString(FIELD_MSG));
				}
				return jsonObject;
		}
		public List<JobInfo> listJobs(ListJobParam param) throws ClientProtocolException, 
IOException, BioflowException,
					InvalidKeyException, NoSuchAlgorithmException, 
InvalidAlgorithmParameterException, NoSuchPaddingException,
					IllegalBlockSizeException, BadPaddingException {
				List<JobInfo> result = null;
				String method = RestInvokerUtils.Method.GET.name();
				String uri = URI_LIST_JOB;
				String url = PROTOCOL + domain + uri;

				// 设置header
				Map<String, String> headers = new HashMap<String, String>();
				headers.put(HEADER_CONTENT_TYPE, CONTENT_TYPE_JSON);
				headers.put(HEADER_X_XTAO_ACCOUNT, getEncryptAccHeader());
				String signatureString = getAuthorizationHeader(method, uri, headers);
				String authorizationHeader = String.format("APIKey=%s,Signature=%s", this.apiKey, 
signatureString);
				headers.put(HEADER_AUTHORIZATION, authorizationHeader);

				// 设置body
				String body = "{}";
				if (param != null) {
					body = JSON.toJSONString(param);
				}
```

```
                // 发送请求
                String responseStr = RestInvokerUtils.invoke(method, url, null, headers, body);

                // 解析结果
                JSONObject jsonObject = responseParse(responseStr);
                JSONArray jsonArray = jsonObject.getJSONArray(FIELD_JOBS);
                result = jsonArray.toJavaList(JobInfo.class);

                return result;
        }
}
```