

# Driving IBM Z Workload Scheduler with REST API

Federica Gari – IBM Z Workload Scheduler Tech Sales  
HCL Technologies

November 2020  
Session 3AF



# Agenda

- IBM ZWS brief overview
- REST API introduction
- IBM ZWS REST API
- Customer scenarios
- Demo

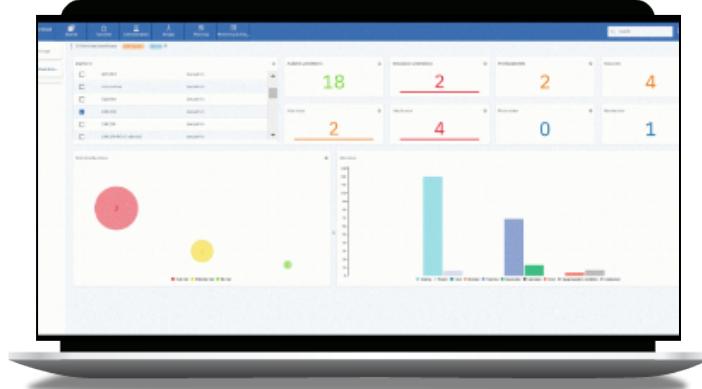
# IBM Z Workload Scheduler – a digital transformation booster

## Automate & De-Risk

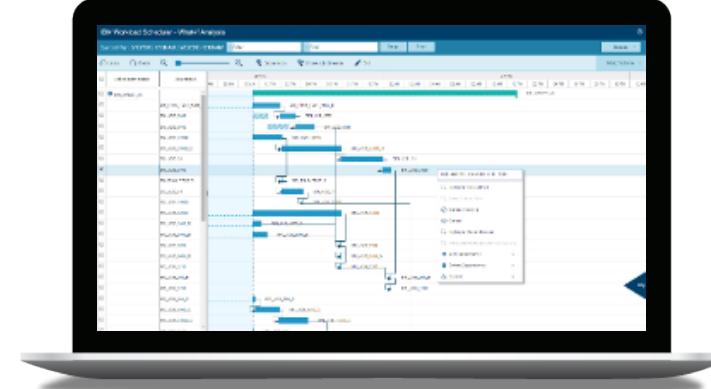


Automation Hub

## Monitor



## Govern

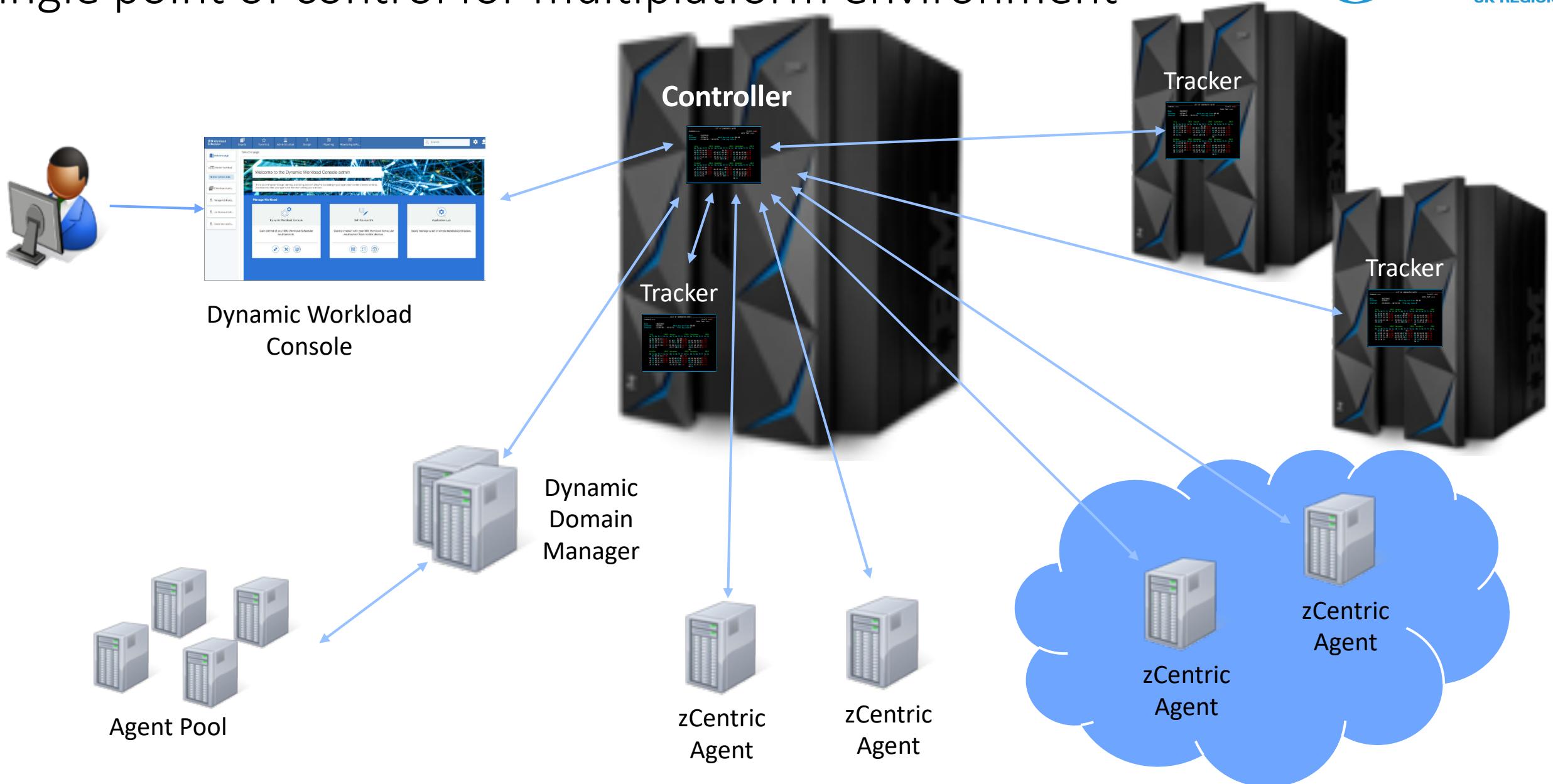


- ✓ Any kind of **unattended** process
- ✓ **Millions** of jobs per day

- ✓ Unified management platform for **advance monitoring and recovery** across all data centers

- ✓ Centralized point for regulation **compliance**
- ✓ **Predicts** impacts of failures on business deadlines with analytics

# Single point of control for multiplatform environment

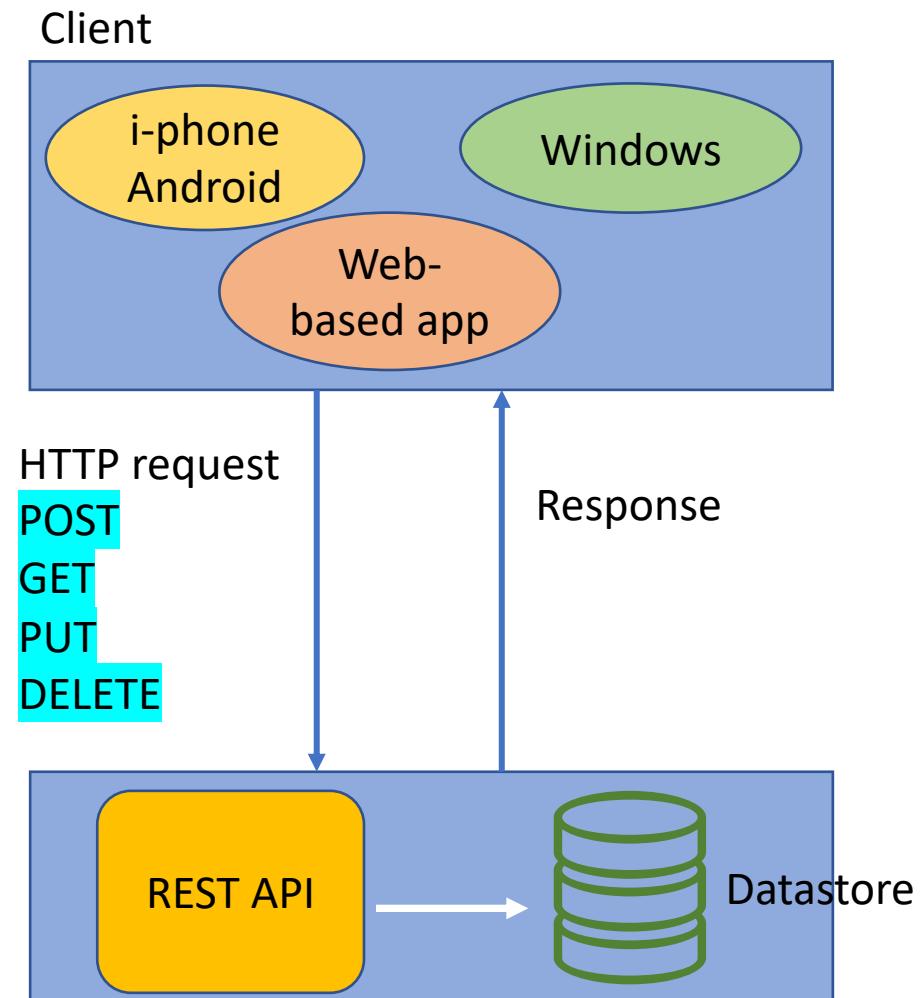


# REST API – Representational State Transfer

A REST API 'is a architectural style' based on web standard and HTTP protocol, that represents the state of a resource at a certain time

- Everything is a resource.
- Resources are identified by global id (URI URL) and have different representation: text , json, xml

A REST server provides access to the resource and a REST client accesses and modifies the resource



# IBM ZWS manages and reduce risks in digital transformation

- Non heterogeneous environments increase complexity of business applications
- Workload increases unexpectedly

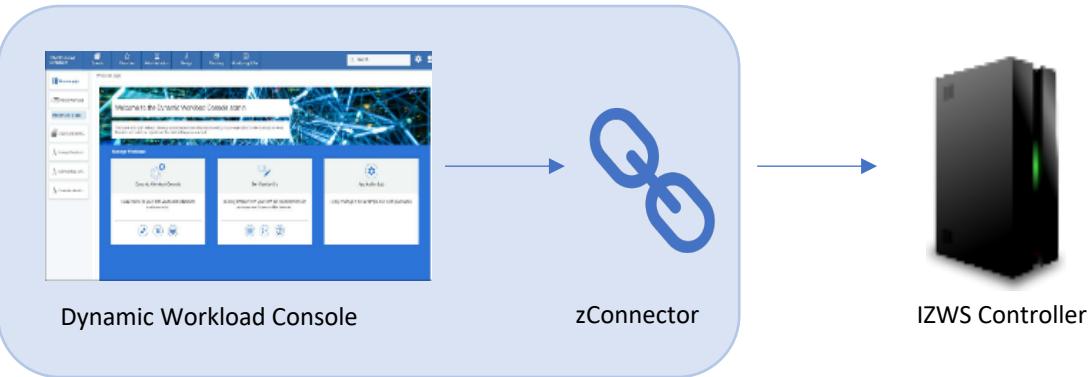


IBM ZWS exposes multiple interfaces to manage business applications from different platforms and with different types of knowledge

- **REST API**
- **Zowe Command Line** (Session Zowe™ Overview and IBM Z Workload Scheduler Zowe extensions - 4AC)
- **IZWS Dashboard**
- **Workload Automation Programming Language**

# REST API - a digital glue in IBM ZWS

REST API for IBM ZWS needs the zOS Connector to the Controller



- Simple web-based API
- Leverage HTTP HTTPS protocol
- Can be called from anywhere in a hybrid environment
- No complex programming

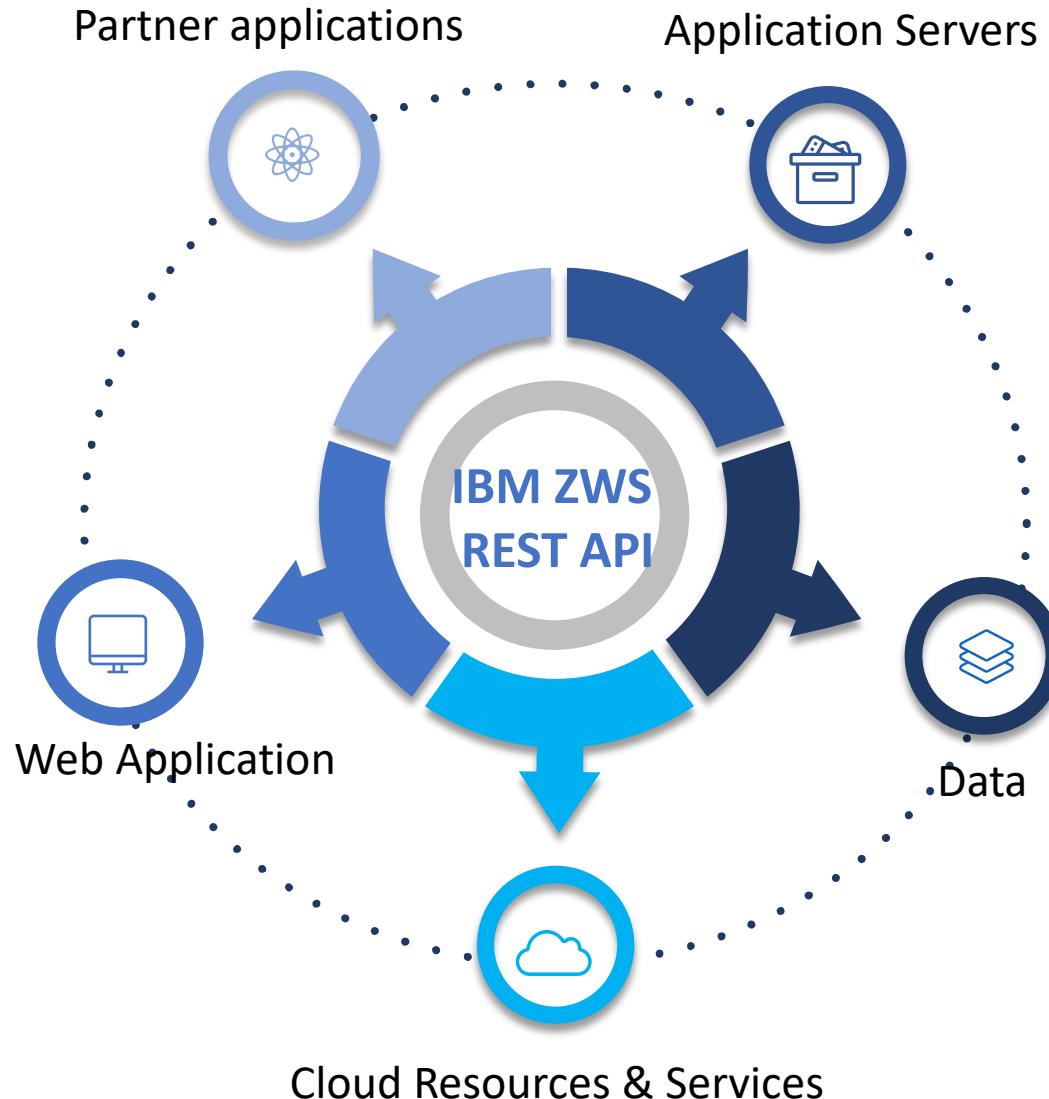
## Acting on IBM ZWS

- Run individual jobs or jobstreams
- Create jobstreams dynamically
- Submit workload at absolute or relative times
- Manipulate active planned workload
- Design or manipulate a resource in the DB

## Third party application acting towards IBM ZWS

- Third party business process information can be embedded within the Dynamic Workload Console Dashboard using their own APIs
- IBM ZWS can run application processes as jobs using their REST APIs

# Get the best of IZWS REST API



IBM ZWS provides a set of fully functional REST APIs. The REST APIs are based on the same APIs used by the product user interface and provide full access to all the IBM Z Workload Scheduler features.

Workload definitions and planned resources can be manipulated using the IBM ZWS REST API. Third party application information can be embedded within the Dynamic Workload Console using their own REST APIs.

# Customer scenarios using REST API

# A Nordic Bank



## Challenge

The customer supports the automation of customers' core financial processes such as Purchase to Pay, Order to Cash and Accounting to Reporting.

- The customer has the need to integrate their ordering system within IBM ZWS, to automatically add applications and jobs instances.

## Solution

The customer is leveraging REST API for IBM ZWS as the fastest and easiest way to accomplish a third-party application integration with IZWS.

- The usage of WAPL REST API, allows them to fully integrate with most of the properties of application/job instances in IBM ZWS plan.

## Benefit

IBM ZWS is the single point of entry to integrate and orchestrate Customer's business applications. This generates cost and process efficiency benefits.



## Challenge

Increase efficiency of IT organization without extensive training

## Solution

The Customer leveraged IBM Z Workload Scheduler (IZWS) REST API in the homegrown Web UI leveraged by developers for other internal purposes to make easy to “submit a job request” providing minimal information.

IZWS REST API were leveraged to add on-demand job to the plan, submit the job added and to retrieve job status.

## Benefit

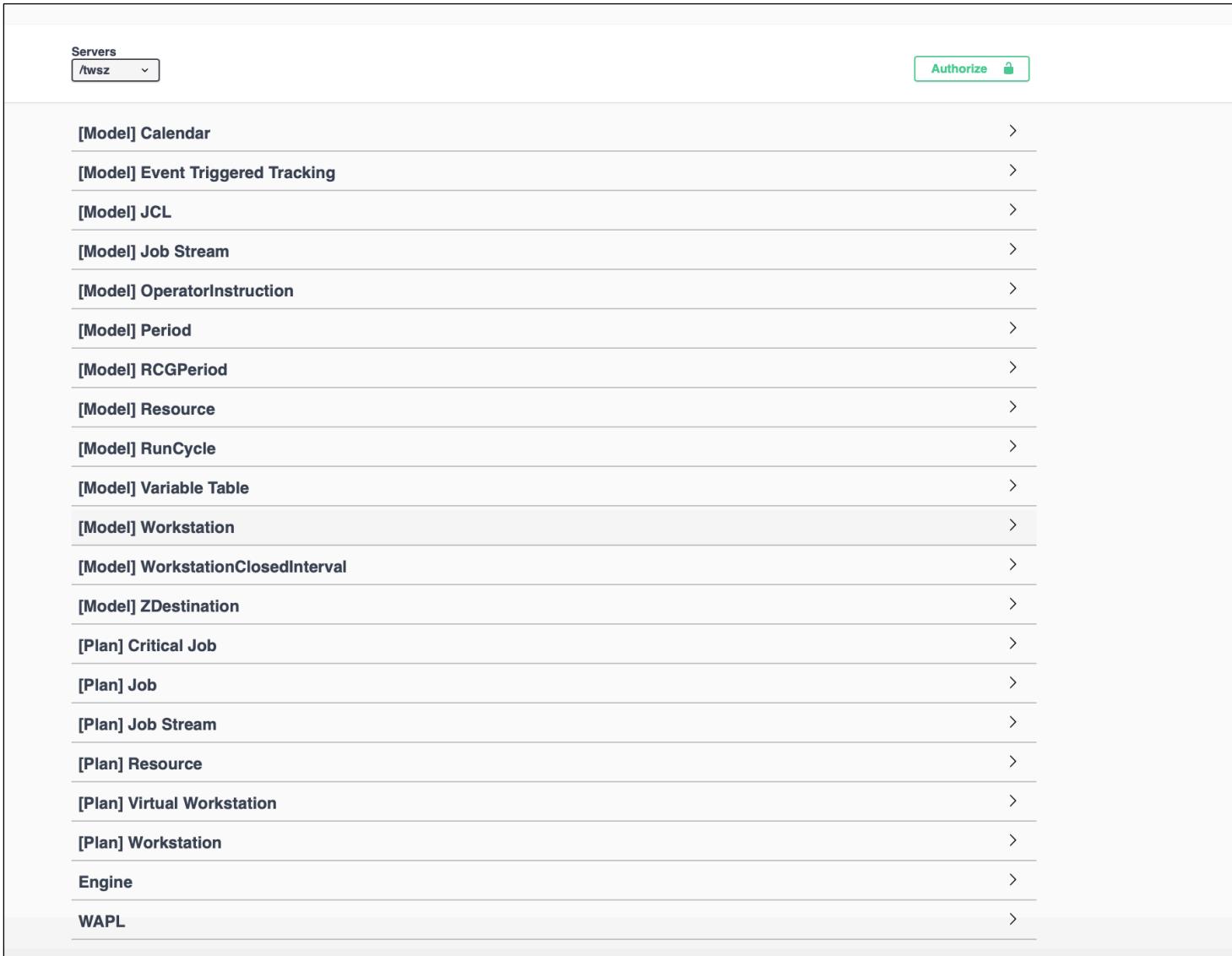
Open scheduling to other teams, with no scheduling skills, in the enterprise :

- Reduce workload on Scheduling team for repetitive task
- Keep under control what is allowed to the end user (through a tailored Web interface and internal check)

# IBM ZWS REST API

# IBM ZWS REST API Swagger

Swagger <https://IP ADDRESS:PORT/twsz/>



The screenshot shows the Swagger interface for the IBM ZWS REST API. At the top left is a dropdown menu labeled 'Servers' with the value '/twsz'. At the top right is a green 'Authorize' button with a lock icon. The main content area is a list of API endpoints, each with a right-pointing arrow to its right. The endpoints are categorized into [Model] and [Plan] sections.

- [Model] Calendar
- [Model] Event Triggered Tracking
- [Model] JCL
- [Model] Job Stream
- [Model] OperatorInstruction
- [Model] Period
- [Model] RCGPeriod
- [Model] Resource
- [Model] RunCycle
- [Model] Variable Table
- [Model] Workstation
- [Model] WorkstationClosedInterval
- [Model] ZDestination
- [Plan] Critical Job
- [Plan] Job
- [Plan] Job Stream
- [Plan] Resource
- [Plan] Virtual Workstation
- [Plan] Workstation
- Engine
- WAPL

3D rules that allows to

- Document
- Design
- Develop

REST API for IBM ZWS

# REST API for modelling

REST API allows to run multiple actions on the DB resources

- Add
- Modify
- Delete
- Query list of resources
- Fetch a resource

Use	Action	HTTP	Body/Filter	Response
Create a new resource	Add	POST	Resource name to be created + some attributes	Resource ID
Update an existing resource	Update	PUT	Resource ID + attributes to be updated	Resource ID
Delete a resource	Delete	DELETE	Resource ID	
Retrieve a list of resources	Query	POST	Can be filtered with attributes	List of resources (ID & name)
Fetch a resource with all attributes	Fetch	GET	Resource ID	Resource with all the attributes

DB Resources

JobStream

Job

Workstation

Calendar

Period

Resource

Variable Table

Run Cycle Group

Event Triggering

Operator Instruction

# REST API for plan

REST API allows to run multiple actions on the Plan resources

- Add
- Modify
- Retrieve list of resources
- Fetch a resource

Use	Action	HTTP	Body/Filter	Response
Create a new resource	Add	POST	Resource name to be created + some attributes	Resource ID
Update an existing resource	Update	PUT	Resource ID + attributes to be updated	Resource ID or No content (for example releasing a job)
Retrieve a list of resources	Query	POST	Can be filtered with attributes	List of resources (ID & name)
Fetch a resource with all attributes	Fetch	GET	Resource ID	Resource with all the attributes

## Plan Resources

**Critical Job** (list, info about risk level and list of predecessors on the critical network)

**Job** (list, submit, update properties, HOLD, NPO, release, set dependencies)

## JobStream

## Joblog

**Dependency** (successor, predecessor, resource, not completed predecessor)

## Resource

## Workstation

# IBM ZWS REST API documentation for the DB

## [Model] Workstation

GET	/v1/{engine}/model/workstation	GET operation for Workstation using a TWSKey	getWorkstationByName	
POST	/v1/{engine}/model/workstation	ADD operation for Workstation	addWorkstation	
DELETE	/v1/{engine}/model/workstation	DELETE operation for Workstation using a TWSKey	removeWorkstationByName	
GET	/v1/{engine}/model/workstation/{workstationId}	GET operation for Workstation using a TWSId	getWorkstationById	

## [Model] Job Stream

GET	/v1/{engine}/model/jobstream	GET operation for JobStream using a TWSKey	getJobStreamByKey	
POST	/v1/{engine}/model/jobstream	ADD operation for JobStream	addJobStream	
DELETE	/v1/{engine}/model/jobstream	DELETE operation for JobStream using a TWSKey	removeJobStreamByKey	
GET	/v1/{engine}/model/jobstream/{jobstreamId}	GET operation for JobStream using a TWSId	getJobStreamById	
PUT	/v1/{engine}/model/jobstream/{jobstreamId}	UPDATE operation for JobStream using a TWSId	updateJobStream	
DELETE	/v1/{engine}/model/jobstream/{jobstreamId}	DELETE operation for JobStream using a TWSId	removeJobStreamById	
PUT	/v1/{engine}/model/jobstream/{jobstreamId}/action/lock	LOCK operation for JobStream using a TWSId	lockJobStreamById	
PUT	/v1/{engine}/model/jobstream/{jobstreamId}/action/unlock	UNLOCK operation for JobStream using a TWSId	unlockJobStreamById	
PUT	/v1/{engine}/model/jobstream/action/lock	LOCK operation for JobStream using a TWSKey	lockJobStreamByKey	
PUT	/v1/{engine}/model/jobstream/action/unlock	UNLOCK operation for JobStream using a TWSKey	unlockJobStreamByKey	
POST	/v1/{engine}/model/jobstream/header/query	QUERY operation for JobStream	queryJobStreamHeader	
POST	/v1/{engine}/model/jobstream/header/query_next	QUERY operation for JobStream	queryNextJobStreamHeader	

# IBM ZWS DB REST API- query a JobStream

QUERY (POST) on JobStream in the DB.  
Filter is on JobStreamName (starting ZOWE)

```
{} response_1603901366604.json x
Users > federica.garihcl.com > Downloads > {} response_1603901366604.json > ...
1  [
2  {
3  "id": "Y29tLmlibS50d3Mu2JqZWN0cy5tb2RlbC5Kb2JTdHJlYW0Awk9XRUFQUAbmYwxzZQBmYwxzZQAyMDIwLjA2LjIzAA==",
4  "key": {
5  "name": "ZOWEAPP",
6  "validFrom": "2020-06-23T00:00:00.000Z",
7  "isDraft": false,
8  "isGroup": false,
9  "className": "com.ibm.tws.objects.model.JobStreamKey"
10 },
11 "description": "Demo Zowe",
12 "priority": 5,
13 "validTo": "2071-12-31T00:00:00.000Z",
14 "carryForward": false,
15 "limit": -1,
16 "groupName": "",
17 "ownerName": "BERRIO",
18 "ownerDescription": "",
19 "authorityGroup": "",
20 "deadlineSmoothingFactor": -1,
21 "deadlineLimitFeedback": -1,
22 "modifyData": {
23   "modifiedBy": "BERRIO",
24   "modifiedTime": "2020-06-23T06:06:00.000Z"
25 },
26 "className": "com.ibm.tws.objects.model.JobStreamHeader"
27 },
28 {
29 "id": "Y29tLmlibS50d3Mu2JqZWN0cy5tb2RlbC5Kb2JTdHJlYW0Awk9XRUpTMDAAZmFsc2UAZmFsc2UAMjAxOS4xMC4yNAA==",
30 "key": {
31   "name": "ZOWEJS00",
32   "validFrom": "2019-10-24T00:00:00.000Z",
33   "isDraft": false,
34   "isGroup": false,
35   "ownerDescription": "ZOWE",
36   "authorityGroup": "",
37   "deadlineSmoothingFactor": -1,
38   "deadlineLimitFeedback": -1,
39   "modifyData": {
40     "modifiedBy": "BERRIO",
41     "modifiedTime": "2020-10-12T11:03:00.000Z"
42   },
43   "className": "com.ibm.tws.objects.model.JobStreamHeader"
44 }
```

```
curl -X POST
"https://10.14.36.57:9443/twsz/v1/cwsusr/model/jobstream/header/query"
-H "accept: application/json" -H "Content-Type: application/json"
-d "{\"filters\":{\"jobstreamFilter\":{\"jobStreamName\":\"ZOWE*\"}}}"
```

```
44   "ownerDescription": "ZOWE",
45   "authorityGroup": "",
46   "deadlineSmoothingFactor": -1,
47   "deadlineLimitFeedback": -1,
48   "modifyData": {
49     "modifiedBy": "BERRIO",
50     "modifiedTime": "2020-10-12T11:03:00.000Z"
51   },
52   "className": "com.ibm.tws.objects.model.JobStreamHeader"
53 }
```

# IBM ZWS REST API documentation for the Current Plan

## [Plan] Job Stream

<b>GET</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}	Get Job Stream In Plan	getJobStreamInPlan	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}	Update JobStream In Plan properties	updateJobStreamInstance	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/cancel	Cancel JobStream In Plan	cancel	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/cancel_pending	Cancel Pending JobStream In Plan	cancelPending	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/complete	Complete a jobstream in plan	complete	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/disable_jobs_monitoring	Disable monitoring of jobs in JobStream by external applications	disableStreamMonitoring	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/enable_jobs_monitoring	Enable monitoring of jobs in JobStream by external applications	enableStreamMonitoring	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/hold	Hold a jobstream in plan	holdJobStream	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/update_jobstream_dependencies	UPDATE a jobstream dependencies in plan	updateJobStreamDependencies	
<b>PUT</b>	/v1/{engine}/plan/{planId}/jobstream/{jobstreamId}/action/wait	Wait a jobstream in plan	setToWaitJobStreamInstance	
<b>POST</b>	/v1/{engine}/plan/{planId}/jobstream/action/add_jobstream	Add operation for Jobstream in plan	addJobStreamInstance	

## [Plan] Workstation

<b>GET</b>	/v1/{engine}/plan/{planId}/workstation/{workstationId}	Get Workstation In Plan	getWorkstationInPlan	
<b>PUT</b>	/v1/{engine}/plan/{planId}/workstation/{workstationId}/action/link	Link workstation In Plan	linkWorkstation	
<b>PUT</b>	/v1/{engine}/plan/{planId}/workstation/{workstationId}/action/status	Set virtual workstation status	setWorkstationStatus	
<b>PUT</b>	/v1/{engine}/plan/{planId}/workstation/{workstationId}/action/unlink	Unlink workstation In Plan	unlinkWorkstation	
<b>PUT</b>	/v1/{engine}/plan/{planId}/workstation/{workstationId}/action/update	Set workstation instance	setWorkstationInstance	
<b>POST</b>	/v1/{engine}/plan/{planId}/workstation/query	Query operation for Workstation in plan	queryWorkstationInPlan	
<b>POST</b>	/v1/{engine}/plan/{planId}/workstation/query_next	Query operation for Workstation in Plan	queryNextWorkstationInPlan	

## [Plan] Critical Job

<b>POST</b>	/v1/{engine}/plan/{planId}/criticaljob/{criticalJobId}/query_predecessors_chain	Predecessors for Critical Job in plan	queryPredecessorsChain	
<b>POST</b>	/v1/{engine}/plan/{planId}/criticaljob/query	Query operation for Critical Job in Plan	queryCriticalJobInPlan	
<b>POST</b>	/v1/{engine}/plan/{planId}/criticaljob/query_next	Query operation for Critical Job in Plan	queryNextCriticalJobInPlan	
<b>POST</b>	/v1/{engine}/plan/{planId}/criticaljob/query_next_predecessors_chain	Next predecessors for Critical Job in plan	queryNextPredecessorsChain	
<b>POST</b>	/v1/{engine}/plan/{planId}/criticaljob/summary_info	Summary info for Critical Job in plan	querySummaryInfo	

# IBM ZWS CP REST API – Add a JobStream in the CP

## ADD (POST) a JobStream (JS1) in the CP

```
curl -X POST
"https://10.14.36.57:9443/twsz/v1/cwsusr/plan/current/jobstream/action/add_jobstream"
-H "accept: application/json" -H "Content-Type: application/json"
-d "{\"name\":\"JS1\"}"
```

{} response\_1603976079452.json ×  
 Users > federica.garihcl.com > Downloads > {} response\_1603976079452.json > ...  
 1 [ ]  
 2 "Y29tLmllibS50d3Mub2JqZWNN0cy5wbGFuLkpvYlN0cmVhbUluUGxhbgBKUzEAMjAyMC4xMC4y0S4xMi41NAA="

## Query (POST) a JobStream (JS1) in the CP

```
curl -X POST
"https://10.14.36.57:9443/twsz/v1/cwsusr/plan/current/jobstream/query"
-H "accept: application/json" -H "Content-Type: application/json"
-d "{\"filters\":{\"jobStreamInPlanFilter\":{\"jobStreamName\":\"JS1*\"}}}"
```

Users > federica.garihcl.com > Downloads > {} response\_1603977237679.json > ...  
 1 [ ]  
 2 {  
 3 "id": "Y29tLmllibS50d3Mub2JqZWNN0cy5wbGFuLkpvYlN0cmVhbUluUGxhbgBKUzEAMjAyMC4xMC4y0S4wMC4xNQA=",  
 4 "description": "",  
 5 "key": {  
 6 "name": "JS1",  
 7 "startTime": "2020-10-29T00:15:00.000Z"  
 8 },  
 48 "status": {  
 49 "commonStatus": "SUCCESSFUL",  
 50 "internalStatus": "COMPLETE",  
 51 "canceled": false  
 52 },  
 53 }

# IBM ZWS CP REST API – Query info on Critical Job

## Query POST of Critical Job information

```
curl -X POST  
"https://10.14.36.57:9443/twsz/v1/cwsusr/plan/current/criticaljob/summary_info"  
-H "accept: application/json" -H "Content-Type: application/json"
```

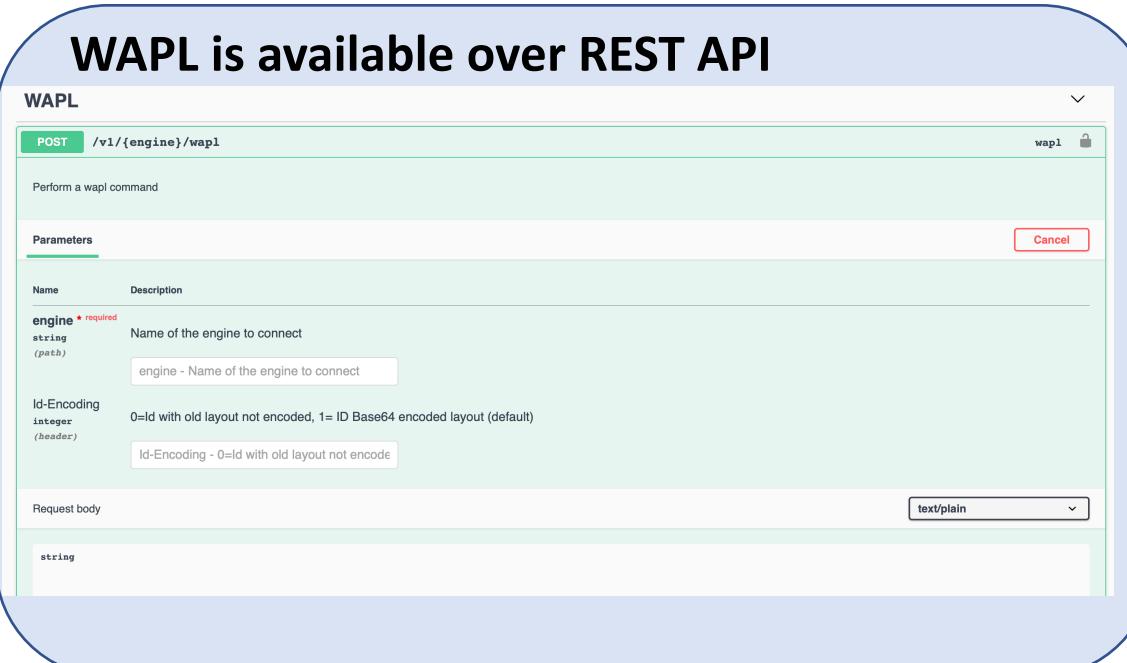
```
{} response_1603979139039.json ×  
  
Users > federica.garihcl.com > Downloads > {} response_1603979139039.json > ...  
1  []  
2  "numberOfCriticalJobsNoRisk": 2,  
3  "numberOfCriticalJobsPotentialRisk": 0,  
4  "numberOfCriticalJobsHighRisk": 3,  
5  "lastCritPathRecalculation": "2020-10-29T01:01:00.000Z"  
6  []
```

# REST API for Workload Automation Programming Language

WAPL a batch utility that allows

- simple commands
- complex programming
- communication with the world outside IBM ZWS

**WAPL is available over REST API**



WAPL

POST /v1/{engine}/wapl

Perform a wapl command

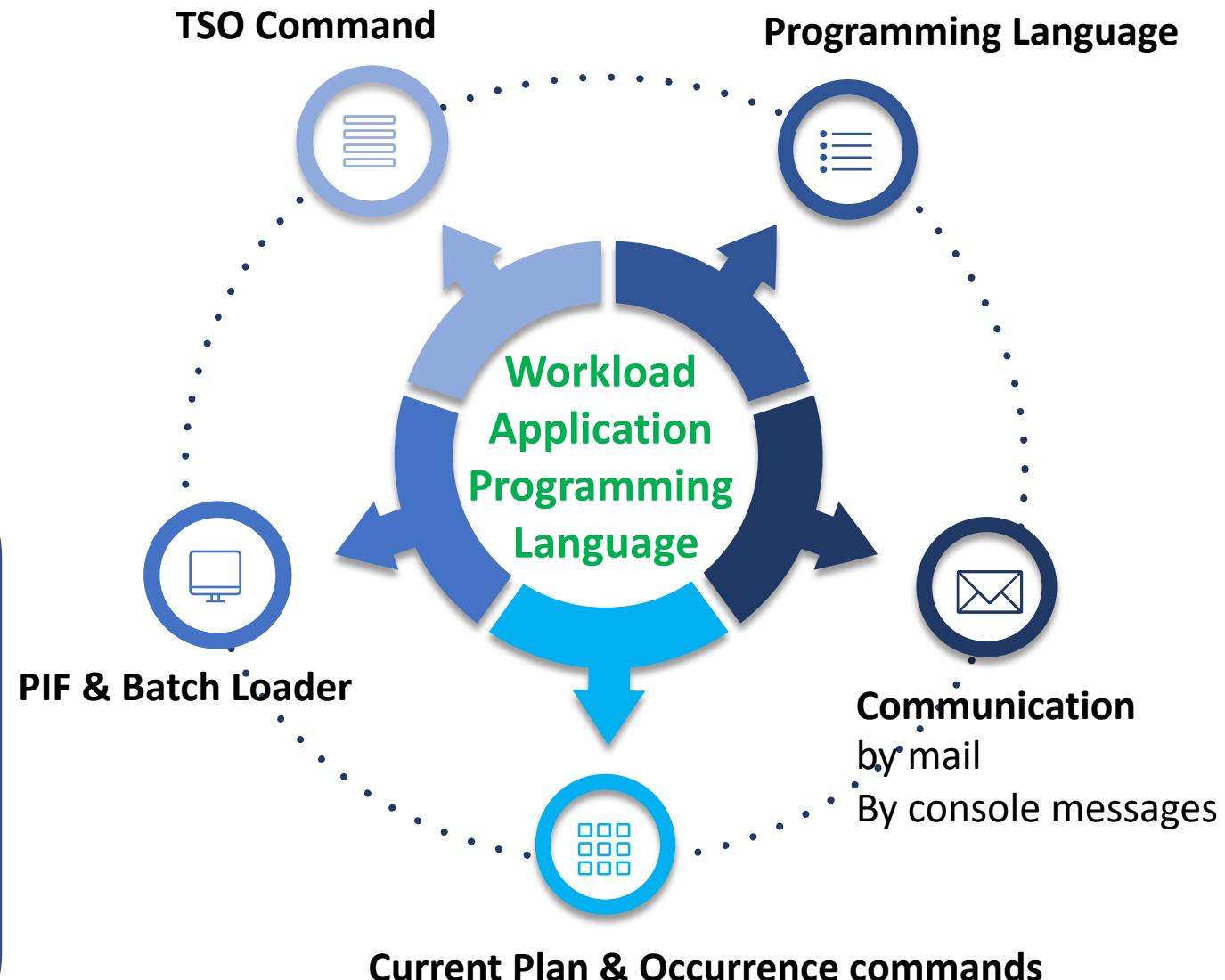
Parameters

Name	Description
engine <small>• required</small>	Name of the engine to connect
Id-Encoding	0=id with old layout not encoded, 1=ID Base64 encoded layout (default)

Request body

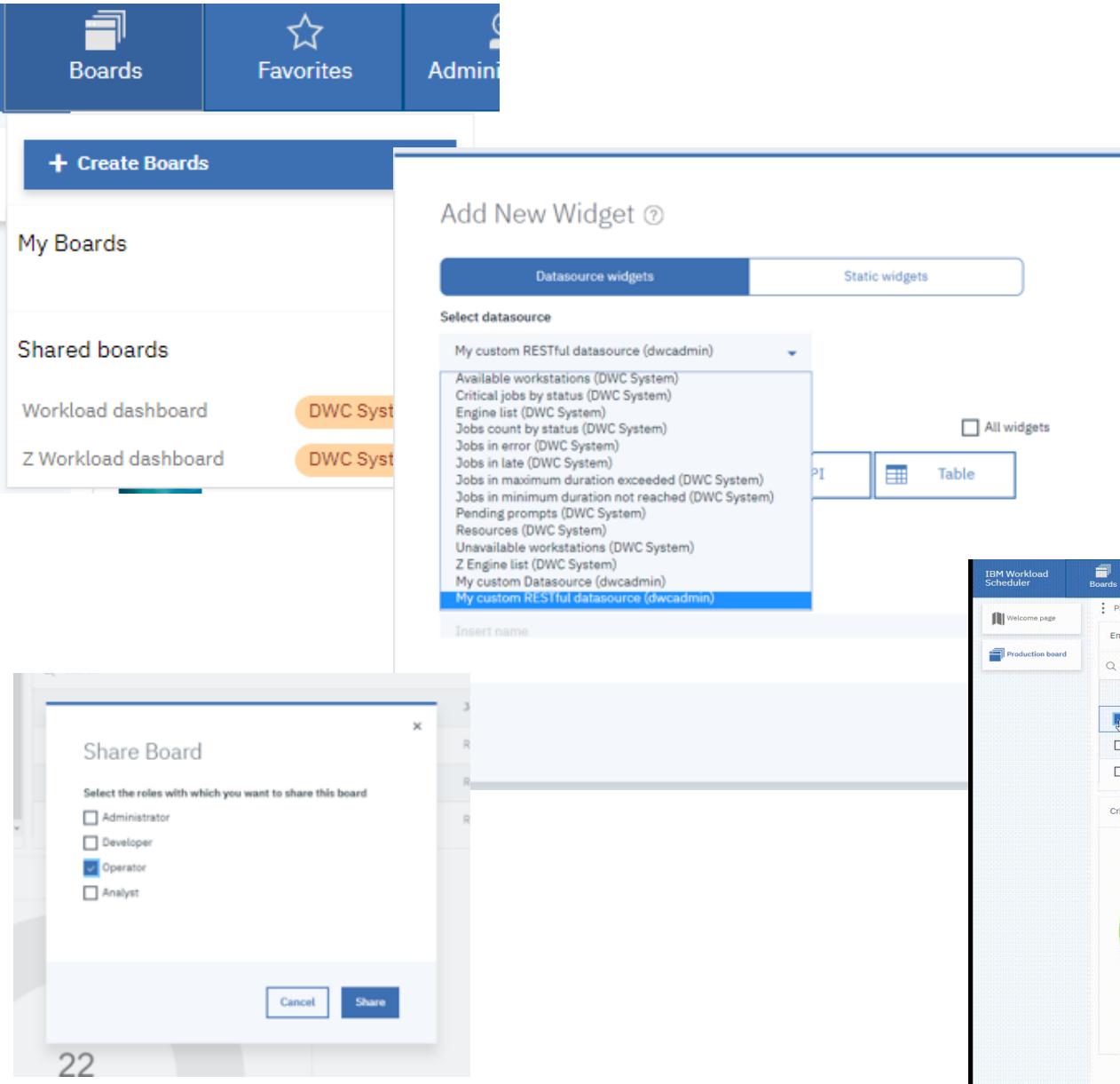
text/plain

string



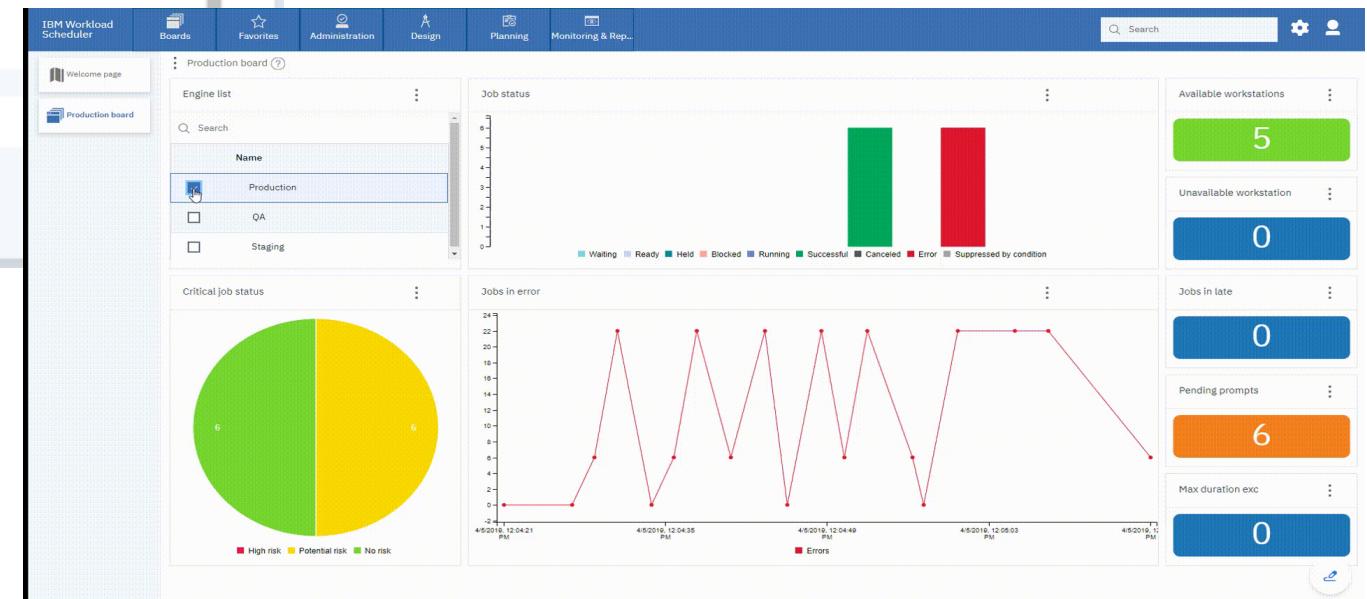
# Dynamic Workload Console Dashboard

# Create your own dashboard



Dashboard can be easily customized

- REST API widget to include IBM ZWS or external application data
- Widget based on IBM ZWS queries
- Static widget to include web page



# Customizable Dashboard

**Datasource name**  
Must be between 1 and 52 characters  
Vienna weather

**Refresh time (seconds)**  
Must be between 30 and 1200 s  
30

**Method**  
GET

**Insert URL**  
SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUERY ] ]  
<https://api.weatherbit.io/v2.0/current?city=Vienna,AT&key=cdebd3db2541446993ad7ab06aff0202>

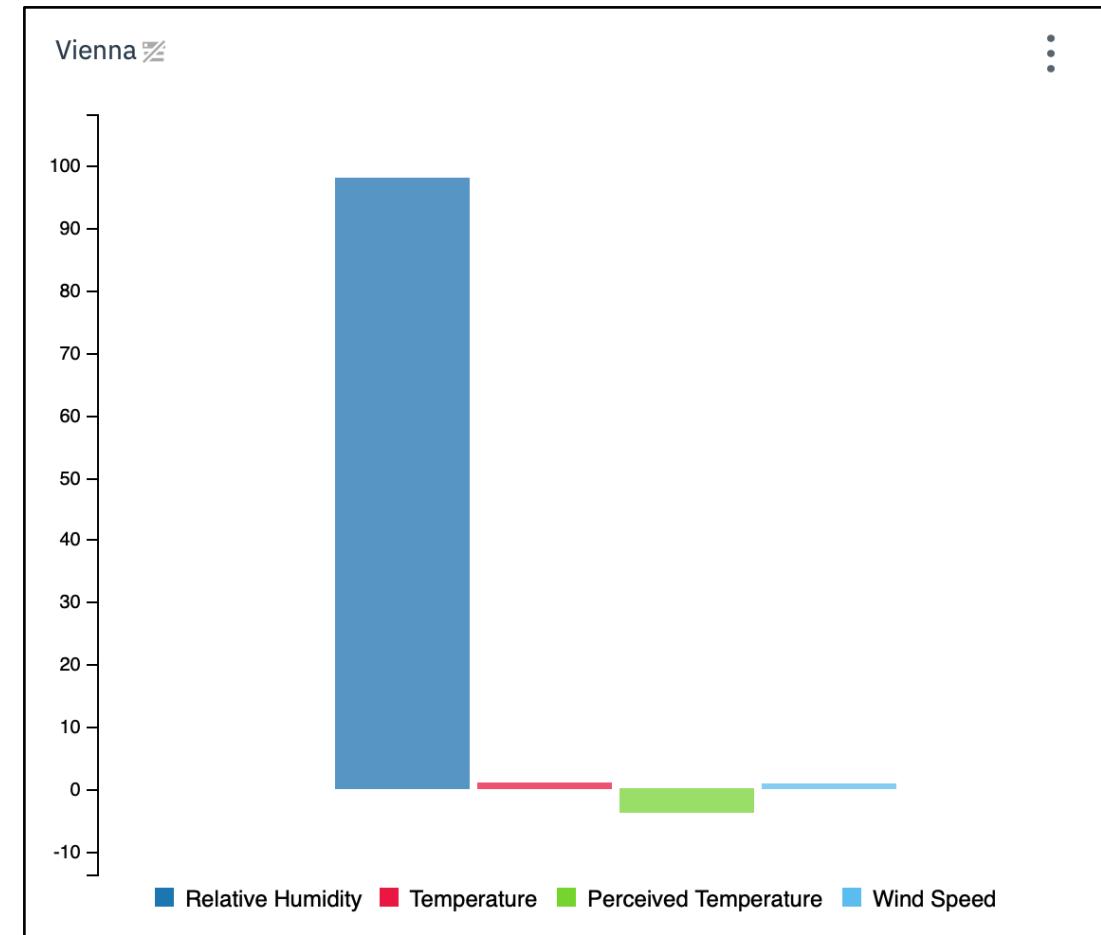
**Headers**  
Accept : application/report Remove row

datasource

Search Add new field

Field	Name	Type	Color
<input type="checkbox"/> data[].rh	Relative Humidity	number	Blue
<input type="checkbox"/> data[].temp	Temperature	number	Red
<input type="checkbox"/> data[].app_temp	Perceived Temperature	number	Green
<input type="checkbox"/> data[].wind_spd	Wind Speed	number	Blue

Widget properties



Widget

# Some REST API examples

# REST API - Demo

- Submit application with first operation in HOLD (WAPL REST API). Job release by its ID.
- Submit a job into IZWS plan (WAPL REST API), without application defined in the DB

# JobStream submission with first operation in HOLD

Request body

text/plain

```
OPTIONS SERVER(Y)  
ADD ADID(JS1) HOLD(START)  
EXECUTE
```

```
curl -X POST "https://10.14.36.57:9443/twsz/v1/cwsusr/wapl" -H  
"accept: text/plain" -H "Content-Type: text/plain"  
-d "OPTIONS SERVER(Y)ADD ADID(JS1) HOLD(START)EXECUTE"
```

Monitor Workload (Owner: admin; Engine: cwsusr,z/OS)

Plan Name: current plan

JS1@

Jobs... Dependencies... Set to Complete Set to Waiting Rerun Job Stream View More Actions 

Status ^ Internal Status ^ Job stream ^ Scheduled ^ Priority ^ Late Job ^

Monitor Jobs

Close View

Plan Name: current plan

Dependencies... Set Status... Execute Job Log... Job Stream View Job Run History What-if More Actions 

Status ^ Internal Status ^ Job Number ^ Job ^ Workstation ^ Job stream ^ Status Details ^ Scheduled Time ^ Job Identifier ^ Error Code ^ Time Dependent ^ Earliest Start ^ Planned Start ^ Actual

<input type="checkbox"/> Held	Arriving	1	 JOBA	CPU1	JS1	Held	10/29/20 11:44 AM	No	10/29/20 11:44 AM	10/29/20 11:44 AM
<input type="checkbox"/> Waiting	Waiting	2	 JOBA	CPU1	JS1	Waiting	10/29/20 11:44 AM	No	10/29/20 11:44 AM	10/29/20 11:44 AM

# Release of job in HOLD

```
{ response_1603992444382.json x
Users > federica.garihcl.com > Downloads > { response_1603992444382.json > ...
1 [
2   {
3     "id": "Y29tLmlibS50d3Mub2JqZWN0cy5wbGFuLkpVYkluUGxhbgBKUzEAMjAyMC4xMC4yOS4xMS40NAAxAA==",
4     "name": "1",
5     "description": "",
6     "jobDefinition": {
7       "jobDefinitionInPlanKey": {
8         "workstationInPlanKey": {
9           "name": "CPU1",
10          "originalName": "CPU1",
11          "symId": "CPU1"
12        }
13      },
14      "taskType": "ZOSJOBTASK",
15      "command": false,
16      "internalStatus": "WAITINGFORINPUT"
17    }
18  }
19 ]
```

Get JobID with POST method

```
curl -X POST "https://10.14.36.57:9443/twsz/v1/cwsusr/plan/current/job/query"
-H "accept: application/json" -H "Content-Type: application/json" -d
"{"filters": {"jobInPlanFilter": {"jobStreamName": "JS1", "task": "JOBA", "internalStatusList": ["WAITINGFORINPUT"]}}}"
```

Release JOB by its ID

```
curl -X PUT
"https://10.14.36.57:9443/twsz/v1/cwsusr/plan/current/job/Y29tLmlibS50d3Mub2JqZWN0cy5wbGFuLkpVYkluUGxhbgBKUzEAMjAyMC4xMC4yOS4xMS40NAAxAA%3D%3D/action/release" -H "accept: */*"
```

Plan Name: current plan

JS1@

Jobs... Dependencies... Set to Complete Set to Waiting Rerun Job Stream View More Actions ▾

Status	Internal Status	Job stream	Scheduled Time	Priority	Late Job Stream
<input type="checkbox"/> <input checked="" type="checkbox"/>	Successful	Complete	JS1	10/29/20 12:15 AM 5	
<input type="checkbox"/> <input checked="" type="checkbox"/>	Successful	Complete	JS1	10/29/20 12:54 PM 5	
<input type="checkbox"/> <input checked="" type="checkbox"/>	Successful	Complete	JS1	10/29/20 11:44 AM 5	

# Submit a job into IZWS plan (WAPL REST API), creating a dynamic application

Request body text/plain

```
OPTIONS SERVER(Y)
ADDJOB IEFBR14 PFX(GSE#JB#)
EXECUTE
```

```
curl -X POST "https://10.14.36.57:9443/twsz/v1/cwsusr/wapl" -H
"accept: text/plain" -H "Content-Type: text/plain"
-d "OPTIONS SERVER(Y)ADDJOB IEFBR14 PFX(GSE#JB#)EXECUTE"
```

## Monitor Workload (Owner: admin; Engine: cwsusr,z/OS)

Plan Name: current plan

GSE@ Run Edit

Jobs... Dependencies... Set to Complete Set to Waiting Rerun Job Stream View More Actions Filter ...

Status ^ Internal Status ^ Job stream ^ Scheduled Time ^ Priority ^ Late Job Stream ^

Successful  Complete GSE#JB#IEFBR14 10/29/20 1:52 PM 5

# Please submit your session feedback!

- Do it online at <http://conferences.gse.org.uk/2020/feedback/3AF>
- This session is **3AF**

1. What is your conference registration number?

💡 This is the three digit number on the bottom of your delegate badge

2. Was the length of this presentation correct?

💡 1 to 4 = "Too Short" 5 = "OK" 6-9 = "Too Long"

1	2	3	4	5	6	7	8	9
<input type="radio"/>								

3. Did this presentation meet your requirements?

💡 1 to 4 = "No" 5 = "OK" 6-9 = "Yes"

1	2	3	4	5	6	7	8	9
<input type="radio"/>								

4. Was the session content what you expected?

💡 1 to 4 = "No" 5 = "OK" 6-9 = "Yes"

1	2	3	4	5	6	7	8	9
<input type="radio"/>								

Place your  
custom session  
QR code here.  
Please remove  
the border and  
text beforehand.

# GSE UK Conference 2020 Charity

- The GSE UK Region team hope that you find this presentation and others that follow useful and help to expand your knowledge of z Systems.
- Please consider showing your appreciation by kindly donating a small sum to our charity this year, NHS Charities Together. Follow the link below or scan the QR Code:

<http://uk.virginmoneygiving.com/GuideShareEuropeUKRegion>

