

Introduction

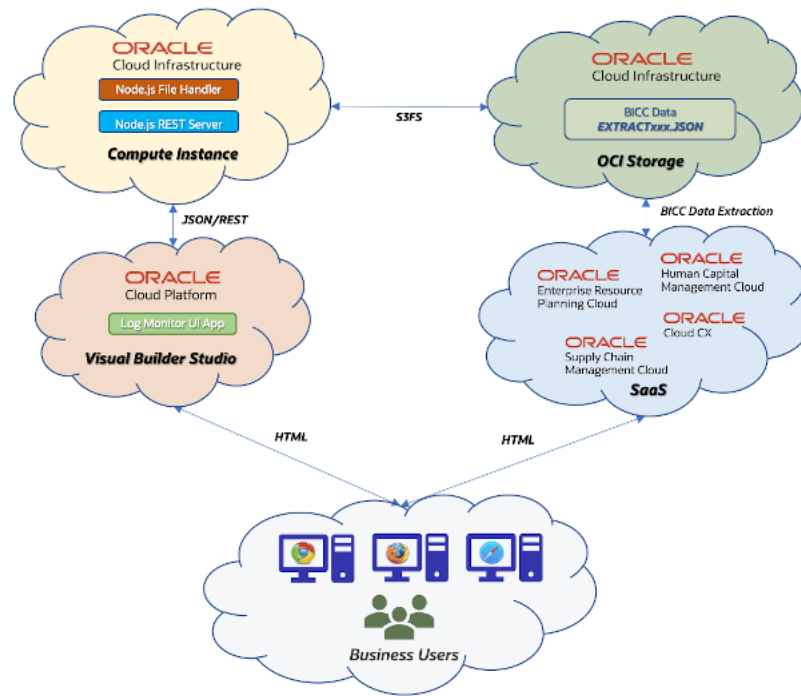
In one of my former [blogs](#) I've written about a mechanism to download and parse the EXTRACT JSON log files via various scripts. If you are not familiar with this concept then its strongly recommended to start with reading that blog first. Also before entering the explanation of the proposed new approach lets conclude what the previous blog was offering as a solution:

- EXTRACT JSON files were downloaded by a Shell script from UCM and stored locally on a server or alternately on a cloud resource
- Via a Node.js sample application described inside that article, all existing EXTRACT JSON files were read and the data were consolidated in memory
- These data were rendered as HTML by this sample application and visualized via an Koa.js Web Server accessible in a browser
- That specific approach had no built-in functionality to refresh the shown data - for a refresh another download of data and program restart was required
- Last but not least the solution uses no components that will require an additional licensing from Oracle side as BICC and UCM in Oracle SaaS are free of charge - all the mentioned other components are available as public Third Party Software

This blog is used to provide an alternate solution that will solve some of the limitations mentioned above, but also introduce an new approach that will allow

- a more dynamic reading from the BICC output directory
- using the capabilities to read the content of this BICC output directory again without stopping the Koa.js service
- the creation of a high sophisticated UI in any tool that is able to consume REST services
- using the recent OCI services to store BICC output and to run the Node.js application in the cloud

As we will use OCI native components here, the suggested solution requires a subscription to the services used in the proposed architecture.



Architecture Overview

BICC in SaaS can be configured to store the extract files on an **OCI Object Storage**. We have already some existing A-Team Chronicles articles like [this](#) that describe the according configuration. In opposite to UCM the files on OCI Object Storage can be used like in a file system while UCM is a document container. That means we need no dedicated script to allow us accessing these data in a file system once downloaded. OCI Object Storage has an AWS S3 compatibility and a storage bucket can be mounted as a remote file system in a Linux machine. This [article](#) provides more details how to mount a bucket as a remote file system.

Once configured in BICC the EXTRACT JSON files will be located in that bucket together with the MANIFEST files and the ZIP files containing the extracted data. These files can be processed from the **Node.js** application running in a Linux VM and being installed on an **OCI**

Compute Instance. This bucket must be mounted as a remote file system via **S3FS**. The sample application will read all the JSON files from the mounted file system and process the information in a similar way as provided in my previous [blog](#). The main changes are basically by providing new features to refresh data, the availability of a set of REST API's and the elimination of HTML rendering. The dedicated views on these data will be provided by REST API's via **Koa.js**.

These REST API's can be used for multiple other purposes like feeding of Analytical Applications, embedding into Integration Flows or using for Data Visualization. As a sample for Data Visualization I have added a brief section with screenshots that are suggesting how to use these services in an **Oracle Visual Builder (VB)** environment - mainly as an alternative to the static HTML rendering like shown in my previous blog. Providing a full-blown UI sample in VB Studio would exceed the volume of a blog like this. In case there is a significant interest in such a blog writing then leave an according comment at the end of this page please!

In the following sections I'll explain the single components more in detail including some screenshots. For all of the exposed components we have plenty of other documentation - blogs, standard documentation or even an OCI certification process will help understanding the details. In the sense to keep focusing on the essentials - the end to end flow from BICC to visualizing the EXTRACT JSON log files - I've kept the further explanations as short as possible. The definite intention of this article is to give some guidelines of an according configuration for customers/partners who'd wish to setup a similar environment.

Configuration of OCI Object Storage and Creation of a Bucket

As a first step in the configuration process we have to create an OCI Object Storage caused by the fact that we need these connection information later when configuring the External Storage in BICC. The following activities give a high level overview of the steps being required to create a bucket that can be used as a remote container in BICC. As shown in screenshot below this configuration starts with an OCI Compartment where the OCI Object Storage will reside in. For our sample it's sufficient to create a "plain" [Compartment](#) as documented on [Oracle Docs](#). The goal is to create a Bucket in the [Object Storage](#) as explained in this [document](#).

ORACLE Cloud Applications > Search for resources, services, and documentation US East (Ashburn) v

Identity » Compartments » Compartment Details

ujanke_compartment_compartment_1566610201042

Compartment created by A-team automated script

Rename Compartment Edit Description Move Resource **Delete** Add Tags

Compartment Information Tags

Parent Compartment: [ateamsaas \(root\)](#)

OCID: ...svzrzq [Show](#) [Copy](#)

Authorized: Yes

Created: Tue, Jun 16, 2020, 14:11:27 UTC

Security Zone: Not Enabled ⓘ

Resources

[Child Compartments \(1\)](#)

[Work Requests \(0\)](#)

[Tag Defaults \(0\)](#)

Filters

STATE

Active | Deleting

Child Compartments

Create Compartment

Name	Status	OCID	Authorized	Security Zone ⓘ	Created
ujanke_compartment	Active	...apdydq	Yes	Not Enabled	Wed, Jul 8, 2020, 13:12:13 UTC

Showing 1 item < Page 1 >

Terms of Use and Privacy Cookie Preferences

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

OCI Compartment definition

Once an Object Storage is accessible, the next step is to create a Bucket to be used for BICC extracts. The detailed documentation for the creation of a Bucket can be found [here](#). The visibility of this Bucket is defined as **Private** here, as the BICC extracts will very likely contain sensible data. Its in the customers responsibility to define this privacy scope!

Oracle Cloud Applications Search for resources, services, and documentation US East (Ashburn)

Object Storage

Object Storage

Data Transfer - Import

Data Transfer - Export

List Scope

COMPARTMENT

ujanke_compartment

allanmccar (root)/ujanke_compartment_compartme
nt_1656610201042/ujanke_compartment

Tag Filters [add](#) | [clear](#)

no tag filters applied

Buckets in ujanke_compartment *Compartment*

Object Storage provides unlimited, high-performance, durable, and secure data storage. Data is uploaded as objects that are stored in buckets. [Learn more](#)

[Create Bucket](#)

Name	Storage Tier	Visibility	Created
ujanke_bucket-20200708	Standard	Private	Wed, Jul 8, 2020, 13:25:01 UTC

Showing 1 item < 1 of 1 >

[Terms of Use and Privacy](#) [Cookie Preferences](#)


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

OCI ObjectStorage Overview - Bucket configured

The overview below provides some details about the actual **Bucket** configuration including the parameter **OCID** as we will need that value in the next step. Its required in BICC to setup this value during the configuration of the external storage. In the screenshot below we can also see how the extraction files are listed in this container.

ORACLE Cloud Applications > Search for resources, services, and documentation US East (Ashburn)

Object Storage > Bucket Details



ujanke_bucket-20200708

Edit Visibility Move Resource Re-encrypt Add Tags Delete

Bucket Information Tags

Visibility: Private
Namespace: ateamswas
Storage Tier: Standard
Approximate Count: 5,024 objects
Etag: 5a39f588-4385-4a86-916c-fc7a4a755ba3
OCID: ...ovkqwas Show Copy

Encryption Key: Oracle managed key [Assign](#)
Created: Wed, Jul 8, 2020, 13:25:01 UTC
Compartment: [ujanke_compartment](#)
Approximate Size: 268.27 MiB
Emit Object Events: Enabled Edit
Object Versioning: Disabled Edit

Resources

- Objects**
- Metrics
- Pre-Authenticated Requests
- Work Requests
- Lifecycle Policy Rules
- Replication Policy
- Retention Rules
- Logs

Objects

Upload More Actions Search by prefix

<input type="checkbox"/>	Name	Last Modified	Size	Status
<input type="checkbox"/>	EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663072.JSON	Mon, Jul 13, 2020, 10:11:23 UTC	6.41 KIB	Available
<input type="checkbox"/>	EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663082.JSON	Mon, Jul 13, 2020, 12:02:18 UTC	6.37 KIB	Available
<input type="checkbox"/>	EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663116.JSON	Mon, Jul 13, 2020, 14:02:26 UTC	6.37 KIB	Available
<input type="checkbox"/>	EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663152.JSON	Mon, Jul 13, 2020, 16:02:13 UTC	6.37 KIB	Available
<input type="checkbox"/>	EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663189.JSON	Mon, Jul 13, 2020, 18:02:19 UTC	6.37 KIB	Available
<input type="checkbox"/>	EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663226.JSON	Mon, Jul 13, 2020, 20:02:13 UTC	6.37 KIB	Available

Terms of Use and Privacy Cookie Preferences Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

OCI ObjectStorage - Bucket Details

Once this initial storage configuration has been done in OCI, we are ready to configure the external storage in BICC in order to save the output of those extraction runs in that Bucket.

Configuring BICC to use OCI Object Storage

In Oracle SaaS we have to create the remote container as an External Storage inside the BICC Console. For less experienced users this [BICC documentation](#) would be a good starting point to understand the usage of BICC.

A privileged user with a role as BICC Admin assigned has to open the **Configure External Storage** icon in order to open the page shown below. Such a new configuration can be added by using the menu item **OCI Object Storage Connection** as being available on that page.

ORACLE Oracle Business Intelligence Cloud Connector Console

Configure External Storage

Storage Service Connection OCI Object Storage Connection JCM Connection

View + x Detail

Name	Host	Namespace	Bucket	Actions
OAC_REPLICAT...	objectstorage.us-ashburn-1.oraclecloud.com	atreamsaa	OVC_REPLICAT...	⚙
OCI_FAW_BUC...	objectstorage.us-ashburn-1.oraclecloud.com	atreamsaa	bucket-faw	⚙
upenke_oci_stor...	objectstorage.us-ashburn-1.oraclecloud.com	atreamsaa	upenke_bucket-...	⚙

Columns Hidden 1

BICC external storage definition 1 of 2

The single configuration items for an OCI Object Storage connection have been documented [here](#). As mentioned [earlier](#) in this article there is also a blog existing on A-Team Chronicles which shares more details.

ORACLE

Oracle Business Intelligence Cloud Connector Console

HelpCASEY.BROWN

Configure External Storage

SaveCancel

OCI Parameters

Name

ujanka_oci_storage

Host

objectstorage.ocicloud.com

Tenancy OCID

ocid1.tenancy.oc1..aaaaa.....fcdm2ygs88k

User OCID

ocid1.user.oc1..aaaaaaa5aqr.....khrnccapa

Namespace

atcam2as

Bucket

ujanka_bucket-20200708

OCI API Signing Key

Generate API Signing Key

Export Public Key

Fingerprint

44:95:00.....5fd8

Note: Make sure to export and add Public Key to OCI before Test Connection.

Verification

Test Connection

BICC external storage definition 2 of 2

Once configured, the external storage can be chosen as the output target - either as a default value or for dedicated extracts. The entire output of an extraction run will be located in that target: the zip files containing the data, the MANIFEST files with the metadata and the EXTRACT JSON files with the run statistics.

ORACLE

Oracle Business Intelligence Cloud Connector Console

HelpCASEY.BROWN

Manage Extract Schedules

Search

Show Stateless Schedules

Submission Time after20. 4. 6 04:53 PM

SearchClear

Schedules

ActionsViewCancel ScheduleDelete ScheduleDelete Inactive Schedules

Name	Schedule Id	Recurrence	Submitter	State	External Storage	Global	JobType
AppleTest	1688342	IMMEDIATE	CASEY.BROWN	ERROR	ujarke_oci_storage	—	Application Data and Active Primary Key Extract

Columns Hidden 1

Schedule Requests

ViewView All Runs

Name	Schedule Time	Submission Time	End Time	Request Id	State	Message
AppleTest	20. 8. 3 04:00 PM	20. 9. 3 04:00 PM	20. 9. 4 07:31 AM	1688342	ERROR	ERROR[BACM0104] Cause: BI Data Extract Job failure due to 1 Job timeout.

BICC extract schedule

By finishing the BICC configuration shown above all steps were done to ensure we can use a container in the cloud to store the extraction output for further processing. The next sections will be used to give an overview how this file processing and the REST API consumption was implemented.

Creation and Configuration of a Compute Cloud instance to run a OCI Linux VM

As a base for further processing of the BICC Extract JSON files we need a Compute Instance that will run a Virtual Machine. As you will see further below, we will use *Node.js* and *Koa.js* inside this VM to run the sample application. Useful instructions how to use OCI Compute can be found via the online documentation [here](#). In theory this VM can run a Windows or a Linux OS, but some components like the S3FS package are probably not available under Windows. Honestly this hasn't been checked for this article and so I've used Linux as an OS.

Probably it is also worth to mention that we can use alternate components in this entire solution architecture by choosing a Docker image instead of a Linux VM. As we see, there are multiple choices and also a Docker image would have some advantages for the instance management. Moreover the entire solution is enabled for a CI/CD usage and the hosting VM/Image and OS can be used according to the implementers preferences. However, for this sample, I've chosen to strictly use a Compute Instance that runs a Linux VM.

In the screenshot below you can see some of the parameters being used for the sample Compute Instance. The instance is available from the Internet via its *Public IP Address*. In next section we take a deeper look into the network configuration as being required for our use case.

ORACLE Cloud Applications > Search for resources, services, and documentation US East (Ashburn)

Compute » Instances » Instance Details

ujanke_vm_oel7

Start Stop Reboot Edit More Actions

Instance Information Tags

General Information

Availability Domain: AD-1
 Fault Domain: FD-3
 Region: iad
 OCID: ...cu76cq [Show](#) [Copy](#)
 Launched: Wed, Jul 8, 2020, 13:21:20 UTC
 Compartment: steamsas (root)/ujanke compartment compartment 15686102D1042/ujanke compartment
 Oracle Cloud Agent Management: Enabled ⓘ

Instance Details

Virtual Cloud Network: [ujanke_vcn-20200708](#)
 Maintenance Reboot: -
 Image: [Oracle Linux 7.8-2020.06.30-0](#)
 Launch Mode: NATIVE
 Maintenance Recovery Action: Restore Instance

Shape Configuration

Shape: VM.Standard.E3.Flex
 OCPU Count: 4
 Network Bandwidth (Gbps): 4

Instance Access

You connect to a running Linux instance using a Secure Shell (SSH) connection. You'll need the private key from the SSH key pair that was used to create the instance.

Public IP Address: XXXXXXXXXX [Copy](#)
 Username: opc

Primary VNIC

Private IP Address: 10.0.0.2
 Network Security Groups: None [Edit](#) ⓘ
 Internal FQDN: [ujanke-vm-oel7...](#) [Show](#) [Copy](#)
 Subnet: [Public Subnet](#)

Launch Options

NIC Attachment Type: VFIO
 Remote Data Volume: PARAVIRTUALIZED
 Firmware: UEFI_64
 Boot Volume Type: PARAVIRTUALIZED
 In-transit Encryption: Disabled

[Terms of Use and Privacy](#) [Cookie Preferences](#) Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Overview of OCI Public Cloud Linux VM configuration

Once this instance has been created there are i.e. some typical tasks that are useful before continuing with the next steps as :

- Install the latest OS patches

- Install a Linux user that will own and run the application
- Create a directory to be mounted later as S3FS filesystem
- ...

Any sizing options like number of OCPU's, memory etc. can be handled flexibly according to the estimated volumes of files being processed, the expected/planned number of web sessions, the frequency of data refreshes and similar. The further configuration steps are listed below in detail.

Various other configurations: VCN, S3FS, Node, Koa etc

VCN configurations

Important:

The configuration below reflects an evaluation environment with quite low security restrictions.

This is caused by the fact that we are focusing on an end-to-end scenario with a reduced scope on an enhanced security setup in VCN.

In a real-world production environment these simple settings are not recommended.

More information about networking capabilities to improve the security standards can be found in the [OCI Networking Documentation](#).

This Compute Instance has a **Public IP Address** and the Linux VM has a **Private IP Address**. While the outbound IP traffic is open through NAT, the challenge is to open the inbound traffic too, to allow an access for the REST API's via *http*. The single steps below can be used as a sample configuration path to open the bi-directional network traffic completely to the Linux machine. For this purpose we have to create a VCN configuration as shown below.

Oracle Cloud Applications > Search for resources, services, and documentation US East (Ashburn)

Networking

- Overview
- Virtual Cloud Networks
- Dynamic Routing Gateways
- Customer-Premises Equipment
- VPN Connections
- Load Balancers
- FastConnect
- IP Management
- DNS Zone Management
- TSIG Keys
- Traffic Management Steering Policies
- HTTP Redirects

List Scope

COMPARTMENT

ujanke_compartment

Filters

STATE

Virtual Cloud Networks in ujanke_compartment *Compartment*

Virtual Cloud Networks are virtual, private networks that you set up in Oracle data centers. It closely resembles a traditional network, with firewall rules and specific types of communication gateways that you can choose to use.

Create VCN Start VCN Wizard

Name	State	CIDR Block	Default Route Table	DNS Domain Name	Created
ujanke_vcn-20200708	Available	10.0.0.0/16	Default Route Table for ujanke_vcn-20200708	vcn.oraclevcn.com	Wed, Jul 8, 2020, 13:21:15 UTC

Showing 1 item < 1 of 1 >

Terms of Use and Privacy Cookie Preferences


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Overview of OCI Linux VM network configuration

As shown in the screenshot below there must be a subnet existent for the **Private IP Address** inside the Linux VM. In our test configuration the assigned private IP address is **10.0.0.1** and therefore the configured **CIDR Block** must be **10.0.0.0/24**.

ORACLE Cloud Applications Search for resources, services, and documentation US East (Ashburn)

Networking > Virtual Cloud Networks > Virtual Cloud Network Details



ujanke_vcn-20200708

Move Resource Add Tags Terminate

VCN Information Tags

Compartment: ujanke_compartment
Created: Wed, Jul 8, 2020, 13:21:15 UTC
CIDR Block: 10.0.0.0/16

OCID: ...wbzdbq Show Copy
Default Route Table: [Default Route Table for ujanke_vcn-20200708](#)
DNS Domain Name: vcn.oraclecloud.com

AVAILABLE

Resources

Subnets (1)

Route Tables (2)

Internet Gateways (1)

Dynamic Routing Gateways (0)

Network Security Groups (0)

Security Lists (2)

DHCP Options (1)

Local Peering Gateways (0)

NAT Gateways (0)

Service Gateways (0)

Virtual Machines (0)

Terms of Use and Privacy Cookie Preferences

Subnets in ujanke_compartment Compartment

Create Subnet

Name	State	CIDR Block	Subnet Access	Created
Public Subnet	Available	10.0.0.0/24	Public (Regional)	Wed, Jul 8, 2020, 13:21:17 UTC

Showing 1 item < 1 of 1 >

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Overview of OCI Public Cloud Linux VM configuration

For this evaluation configuration we've configured a **Default Route Table** with a **Route Rule** defining **0.0.0.0/0** as destination and of target type **Internet Gateway**.

ORACLE Cloud Applications > Search for resources, services, and documentation US East (Ashburn) 2. ?

Networking > Virtual Cloud Networks > ujanke_vcn-20200708 > Route Table Details

Default Route Table for ujanke_vcn-20200708

Move Resource Add Tags Terminate

RT

AVAILABLE

Resources

Route Rules (1)

Route Table Information Tags

OCID: ...qwjibq Show Copy Compartment: ujanke_compartment

Created: Wed, Jul 8, 2020, 13:21:15 UTC

Route Rules

Add Route Rules Edit Remove

<input type="checkbox"/>	Destination	Target Type	Target	Description
<input type="checkbox"/>	0.0.0.0/0	Internet Gateway	Internet Gateway ujanke_vcn-20200708	

0 Selected Showing 1 item < 1 of 1 >

[Terms of Use and Privacy](#) [Cookie Preferences](#) Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Overview of VCN Default Route Table configuration

Another resource to configure are the **DHCP Options**. Here we created a default resource using a DNS Type of **Internet and VCN Resolver** with a standard search domain **vcn.oraclevcn.com**.

ORACLE Cloud Applications > Search for resources, services, and documentation US East (Ashburn)

Networking > Virtual Cloud Networks > Virtual Cloud Network Details > DHCP Options

ujanke_vcn-20200708

Move Resource Add Tags Terminate

VCN
AVAILABLE

VCN Information Tags

Compartment: ujanke_compartment
Created: Wed, Jul 8, 2020, 13:21:15 UTC
CIDR Block: 10.0.0.0/16

OCID: ...wbzdbq [Show](#) [Copy](#)
Default Route Table: [Default Route Table for ujanke_vcn-20200708](#)
DNS Domain Name: vcn.oraclecloud.com

Resources

DHCP Options in ujanke_compartment Compartment

Create DHCP Options

Name	State	DNS Type	DNS Servers	Search Domain	Created
Default DHCP Options for ujanke_vcn-20200708	Available	Internet and VCN Resolver		vcn... Show Copy	Wed, Jul 8, 2020, 13:21:16 UTC

Showing 1 item < 1 of 1 >

Subnets (1)
Route Tables (2)
Internet Gateways (1)
Dynamic Routing Gateways (0)
Network Security Groups (3)
Security Lists (2)
DHCP Options (1)
Local Peering Gateways (0)
NAT Gateways (0)
Service Gateways (0)
VNICs (1)

[Terms of Use and Privacy](#) [Cookie Preferences](#)

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Overview of VCN DHCP options configuration

In another configuration step we have to create an **Internet Gateway** that will be used as a target for a new **Route Rule** to be created. The Internet Gateway has no further configuration options. The destination for this Route Rule must be **0.0.0.0/0** with a target for the previously created Internet Gateway. This will be used to redirect the outbound and inbound IP traffic between private and public networks.

The screenshot displays the Oracle Cloud console interface. At the top, the navigation bar includes the Oracle Cloud logo, a search bar, and the region 'US East (Ashburn)'. The breadcrumb trail indicates the path: 'Networking > Virtual Cloud Networks > ujanke_vcn-20200708 > Route Table Details'.

The main content area is titled 'Default Route'. On the left, there is a green hexagonal icon with 'RT' and the status 'AVAILABLE'. To the right, a modal window provides details for the 'Internet Gateway ujanke_vcn-20200708':

- Name:** Internet Gateway ujanke_vcn-20200708
- Status:** Enabled
- Compartment:** ujanke_compartment
- OCID:** ...gle00a [Show](#) [Copy](#)
- State:** ● Available
- Created:** Wed, Jul 8, 2020, 13:21:16 UTC

Below the modal, the 'Route Table Information' section shows the OCID and creation time for the route table. The 'Route Rules' section features a table with one rule:

Destination	Target Type	Target	Description
0.0.0.0/0	Internet Gateway	Internet Gateway ujanke_vcn-20200708	

The table includes controls for adding, editing, and removing rules, and a status bar at the bottom indicates 'Showing 1 item'.


Route Rule for a VCN Internet Gateway configuration

Once the Internet Gateway and Routing Rules are existent we must create in another step a ***Security List*** (in screenshot below named ***ujanke_http_virt_server***) that defines the security rules for the ***http*** communication we intend to run. This Security List will contain rules for inbound (***Ingress***) and outbound (***Egress***) IP traffic.

ORACLE Cloud Applications > Search for resources, services, and documentation

Networking > Virtual Cloud Networks > ujanke_vcn-20200708 > Subnet Details

Public Subnet

 AVAILABLE

[Edit](#) [Move Resource](#) [Add Tags](#) [Terminate](#)

Subnet Information **Tags**

OCID: [...oyu7xq](#) [Show](#) [Copy](#)
CIDR Block: 10.0.0.0/24
Virtual Router Mac Address: 00:00:17:BA:CE:19
Subnet Type: Regional

Compartment: ujanke_compartment
DNS Domain Name: subnet... [Show](#) [Copy](#)
Subnet Access: Public Subnet
DHCP Options: [Default DHCP Options for ujanke_vcn-20200708](#)
Route Table: [Default Route Table for ujanke_vcn-20200708](#)

Resources

Security Lists

[Add Security List](#)

Name	State	Compartment	Created
ujanke_http_virt_server	Available	ujanke_compartment	Wed, Aug 19, 2020, 08:07:33 UTC
Default Security List for ujanke_vcn-20200708	Available	ujanke_compartment	Wed, Jul 8, 2020, 13:21:15 UTC

Showing 2 items < 1 of 1 >

[Terms of Use and Privacy](#) [Cookie Preferences](#) Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Overview of VCN Security Lists configuration

As mentioned further above this test/evaluation configuration has no hard restrictions. If those would be implemented i.e. the Ingress/Egress rules would be used to set tighter boundaries to control the traffic. In our case the both rules are not strict at all and so they both look very

similar. The **Egress Rule** rule is defined as **Not Stateless** with a Destination **0.0.0.0/0** for all protocols and ports. This will set no limitations for the outbound traffic.

The screenshot displays the Oracle Cloud console interface. At the top, the navigation bar includes the Oracle Cloud logo, a search bar, and the region 'US East (Ashburn)'. The breadcrumb trail indicates the path: Networking > Virtual Cloud Networks > ujanke_vcn-20200708 > Security List Details > Egress Rules.

The main content area features a green hexagonal icon with 'SL' and the status 'AVAILABLE' for the resource 'ujanke_http_virt_server'. Below this, there are buttons for 'Move Resource', 'Add Tags', and 'Terminate'. A section titled 'Security List Information' shows the OCID as '...d3y4qa' and the creation date as 'Wed, Aug 18, 2020, 08:07:33 UTC'. The compartment is listed as 'ujanke_compartment'.

A red rectangular box highlights the 'Egress Rules' section. On the left, under 'Resources', 'Egress Rules (1)' is selected. The 'Egress Rules' table has buttons for 'Add Egress Rules', 'Edit', and 'Remove'. The table contains one rule with the following details:

<input type="checkbox"/>	Stateless ▾	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
<input type="checkbox"/>	No	0.0.0.0/0	All Protocols				All traffic for all ports	⋮

At the bottom of the table, it says '0 Selected' and 'Showing 1 item < 1 of 1 >'.

The footer of the page includes links for 'Terms of Use and Privacy' and 'Cookie Preferences', and a copyright notice: 'Copyright © 2020, Oracle and/or its affiliates. All rights reserved.'

Overview of VCN Security Lists Egress Rules configuration

Similar to the Egress Rule above we will set the **Ingress Rule** to the least restrictions on the inbound traffic. As shown below this rule is similar to the Egress Rule we've created above.

The screenshot displays the Oracle Cloud console interface. At the top, the navigation bar includes the Oracle Cloud logo, a search bar, and the region 'US East (Ashburn)'. The breadcrumb trail indicates the path: Networking > Virtual Cloud Networks > ujanke_vcn-20200708 > Security List Details.

The main content area is titled 'ujanke_http_virt_server'. It features a green hexagonal icon with 'SL' and the status 'AVAILABLE'. Below the icon are buttons for 'Move Resource', 'Add Tags', and 'Terminate'. A note states: 'Instance traffic is controlled by firewall rules on each instance in addition to this Security List'.

The 'Security List Information' tab is active, showing the OCID '...d3y4qa' and the compartment 'ujanke_compartment'. The creation date is 'Wed, Aug 19, 2020, 08:07:33 UTC'.

The 'Ingress Rules' section is highlighted with a red border. It contains a table with one rule. The table has columns: Stateless, Source, IP Protocol, Source Port Range, Destination Port Range, Type and Code, Allows, and Description. The rule is named 'No' and allows all traffic for all ports.

<input type="checkbox"/>	Stateless ▼	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
<input type="checkbox"/>	No	0.0.0.0/0	All Protocols				All traffic for all ports	

At the bottom of the console, there are links for 'Terms of Use and Privacy' and 'Cookie Preferences', and a copyright notice for 2020.

Overview of VCN Security Lists Ingress Rules configuration

For our special case of an evaluation purpose it is intended to keep especially the inbound traffic very flexible. It will be controlled from inside our Linux machine by the internal firewall. Via an Node.js and Koa.js application, running a Web Server that will handle REST API calls. The details for this configuration are explained in sections below.

Network configuration in Linux VM

After creating the VCN configuration the firewall in the Linux VM must be opened for the TCP/IP port run by the Web Server. The application will run by default on port 3000, but it can be controlled by a parameter to use another port. It is necessary to open the network port for the application to be accessible from outside the Linux machine. This can be achieved by running a command like this:

```
sudo firewall-cmd --permanent --zone=public --add-port=3000
firewall-cmd --reload
```

Obviously when intending to run the Web Server on another port, then the command above must use the other port number instead of the number **3000**.

If not controlled by a program parameter, the Web Server will run on the local loopback IP address 127.0.0.1. This is not useful in an OCI Compute Instance and so it makes sense to use the private IP address as such a parameter. Alternately and more comfortably it would also make sense to create a hostname inside the Linux VM for the private address in the */etc/hosts*. As shown below an entry would look like this as been taken from our test environment:

```
10.0.0.2 ujanke-vm-oel7.subnet.vcn.oraclevcn.com ujanke-vm-oel7
```

Only by using the private IP address as a program parameter, the Web Server will be reachable from outside as we've configured it in previous VCN setup. The usage of a hostname has some advantages as that will allow scripting which is independent from physical IP addresses.

S3FS configurations

A very important piece in this architecture is the ability to access the BICC output directly in a file system. As shown above, the BICC output files are located in a bucket in an OCI Object Storage. An elegant method to access these files is the ability in OCI Object Storage to provide an AWS S3 compatibility. By installing the Linux package *s3fs-fuse-1.86-2.el7.x86_64* we achieve the capability to mount the remote bucket via S3FS as a local folder. The executable program name is *s3fs*. The local mount point (*/home/myuser/bicc_files* as shown below) must exist or be created before the script below will run successfully.

```
#!/bin/sh

s3fs myuser-bucket /home/myuser/bicc_files \
-o passwd_file=/home/myuser/.passwd-s3fs \
-o url=https://myuser.compat.objectstorage.datacenter.oraclecloud.com \
-o nomultipart \
-o use_path_request_style
```

Details about all the program parameters can be, as usual, obtained by running with the *--help* parameter.

Node.js and Koa.js installation

An additional configuration step inside the Linux VM is the installation of *Node.js* and *Koa.js* before the application can be run. There are multiple ways to install these packages. One option could be *yum* as long as there exists a software repository that provides the required minimum versions: *Node.js 17.4* and *Koa.js 2.13.4*. If these requirements can't be met then an alternate solution could be a better choice:

- Install *npm* as available from <https://nodejs.org>
- If not included then install package *fs* via command *npm install fs --save-bundle*
- Finally install *Koa.js* and depending modules by running
 - *npm install koa --save-bundle*
 - *npm install koa-router --save-bundle*
 - *npm install koa-body --save-bundle*
 - *npm install koa-logger --save-bundle*
 - *npm install koa-static --save-bundle*

Node.js and Koa.js as BICC Log REST Server ...

Once the application has been started there are several REST API's available giving certain views on the BICC extract log data. These are samples and can obviously be modified or extended as the source code is available on GitHub. The sample API's listed below are providing some views on top of the consolidated log data as being read from all the existing EXTRACT JSON files.

- ***reportStatsREST*** -> return the statistics and summaries of all processed files
- ***byDateREST*** -> return all log information ordered by run date
- ***byNameREST*** -> return all log information ordered by VO name
- ***bySchedIDREST*** -> return all log information ordered by Schedule ID
- ***byFailStatusVOREST*** -> return all log information of failed extracts ordered by VO name
- ***byFailStatusDateREST*** -> return all log information of failed extracts ordered by run date
- ***refreshDataREST*** -> initiate another file reading and processing to refresh the REST data

As a built-in online documentation exists an ***index.html*** file that will be displayed when opening the Web Server page in a browser without the virtual path of an existing web service. Here is a sample: the Public IP Address of our Linux VM has been registered locally in a ***hosts*** with a hostname like ***my-bicc-rest-server***. With the URL ***http://my-bicc-rest-server:3000*** in the browser the page below will appear, describing the single REST API's and their JSON payload.

REST API: /reportStatsREST

Operation: GET

Sample Return JSON:

Remark: Element 0 of this JSON structure contains the descriptions the various elements

```
[
  {
    "oldestRunDate": "Oldest Extraction Date/Time",
    "recentRunDate": "Latest Extraction Date/Time",
    "totalNumOfRowsInLogs": "Total Number or Extracted Rows Found",
    "scheduleIDArr": [
      {
        "scheduleID": "List of Batch Schedules Found"
      }
    ],
    "requestIDArr": [
      {
        "requestID": "List of Jobs Found"
      }
    ],
    "fileListArr": [
      {
        "fileName": "Processed File List"
      }
    ],
    "voListArr": [
      {
        "voName": "Processed VO List"
      }
    ]
  },
  {
    "oldestRunDate": "2020-07-13T10:00:06.323Z",
    "recentRunDate": "2020-08-23T18:04:50.747Z",
    "totalNumOfRowsInLogs": "38024907",
    "fileListArr": [
      {
        "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663073.JSON"
      },
      {
        ...
      }
    ]
  }
]
```

Usually the REST API's will be invoked programmatically by tools like *curl* or others. As usual it is also possible to open these URL's in a browser to ensure the target is available or for function teasing and validation purposes. In case the API works as expected and there are data existing you will see a resulting page similar to shown below.

```
▼ [
  ▶ { ... }, // 7 items
  ▼ {
    "oldestRunDate": "2020-07-13T10:00:06.323Z",
    "recentRunDate": "2020-10-05T18:04:55.629Z",
    "totalNumOfRowsInLogs": "635281061",
    ▶ "fileListArr": [ ... ], // 266 items
    ▶ "voListArr": [ ... ], // 415 items
    ▶ "scheduleIDArr": [ ... ], // 2 items
    ▶ "requestIDArr": [ ... ] // 134 items
  }
]
```

Log REST sample JSON Page - overview

The JSON payload consists of a number of arrays containing views on the log data. The very first (zero) element of this structure contains usually some static elements: the array column headers. Those can also be used in the further data visualization tasks as column headers for the visible data. However there is no MLS support with the given approach as these values are in English only. If MLS is a requirement the column headers can't probably be used.

```

▼ [
  ▼ {
    "oldestRunDate": "Oldest Extraction Date/Time",
    "recentRunDate": "Latest Extraction Date/Time",
    "totalNumOfRowsInLogs": "Total Number of Extracted Rows Found",
    ▼ "scheduleIDArr": [
      ▼ {
        "scheduleID": "List of Batch Schedules Found"
      }
    ],
    ▼ "requestIDArr": [
      ▼ {
        "requestID": "List of Jobs Found"
      }
    ],
    ▼ "fileListArr": [
      ▼ {
        "fileName": "Processed File List"
      }
    ],
    ▼ "voListArr": [
      ▼ {
        "voName": "Processed VO List"
      }
    ]
  },
  ▼ {
    "oldestRunDate": "2020-07-13T10:00:06.323Z",
    "recentRunDate": "2020-10-05T10:04:55.629Z",
    "totalNumOfRowsInLogs": "635281061",
    ▼ "fileListArr": [
      ▼ {
        "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1688342_REQUEST_1688342.JSON"
      },
      ▼ {
        "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663073.JSON"
      },
      ▼ {
        "fileName": "EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663072_REQUEST_1663073.JSON"
      },
      ▼ {
        "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663082.JSON"
      },
      ▼ {
        "fileName": "EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663072_REQUEST_1663082.JSON"
      },
      ▼ {
        "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663116.JSON"
      }
    ]
  }
]

```

Log REST sample JSON Page - details

The program collects all data found into JS arrays and creates program structures as separate internal arrays that will be returned as JSON content when doing the various REST calls.

After processing these JSON files a REST server will be started via Koa.js with the following specifications:

- if not specified differently by a parameter the listening port of this server is **3000**
- if not specified differently by a parameter the listing IP address is **127.0.0.1**
- if not specified differently by a parameter there is ***no log output***
- if the given directory is not accessible or can't be read the program execution will be stopped with an according status message
- if no specific REST API is given - means just opening the URL in a browser as is - a static ***index.html*** will be opened and shown providing a documentation about the available services
- if no suitable JSON files exist, but might be produced at a later time, the REST API ***refreshDataREST*** can be used to reread the directory and to rebuild the internal structures
- the ***refreshDataREST*** call can be executed at any time or also scheduled to ensure the latest files will be processed
- none of the REST API's requires a parameter and all are called by using a GET method

The following fields are read and processed as read from the various JSON files:

- *name*
- *status*
- *runDate*
- *queryDurationInSec*
- *extractDurationInSec*
- *uploadDurationInSec*
- *totalDurationInSec*
- *rowCount*
- *errorMessage*

To execute the program run on command line:

```
node biccLogServer.js
```

In case the program has been started like shown above the web site can be opened in a browser on a local by opening the following URL using the loopback address:

```
http://127.0.0.1:3000
```

If running the program without any parameter the caller will receive a standard usage message listening the available paramaters.

```
$ node biccLogServer.js
Error: insufficient number of arguments!
Usage: /usr/local/bin/node /Users/ujanke/Tools/BICC/BICCLogServer/biccLogServer.js <dir>
      [port=] [log=<y|Y>] [host=<server>]
<dir>          -> <Directory containing EXTRACT JSON Log Files>
[port=<value>]  -> web listening port - default value = 3000 [optional parameter]
[log=<y|Y>]     -> console log y|Y - default value = no [optional parameter]
[host=<server>] -> host name or IP addr REST server will listen - default 127.0.0.1
                  [optional parameter]
```

The application will remain running in foreground until being interrupted by Ctrl-C. It has been tested and it works to run the program behind a nohup command on Linux and to put in into the background without interruptig.

If the refresh of log data is a demand it can achieved by running a command like this:

```
curl http://127.0.0.1:3000/refreshDataREST
```

All other API calls in a real-life environment using local hostnames would be executed like this command and retrieve a JSON payload on command line:

```
curl http://my-bicc-rest-server:3000/byNameREST
```

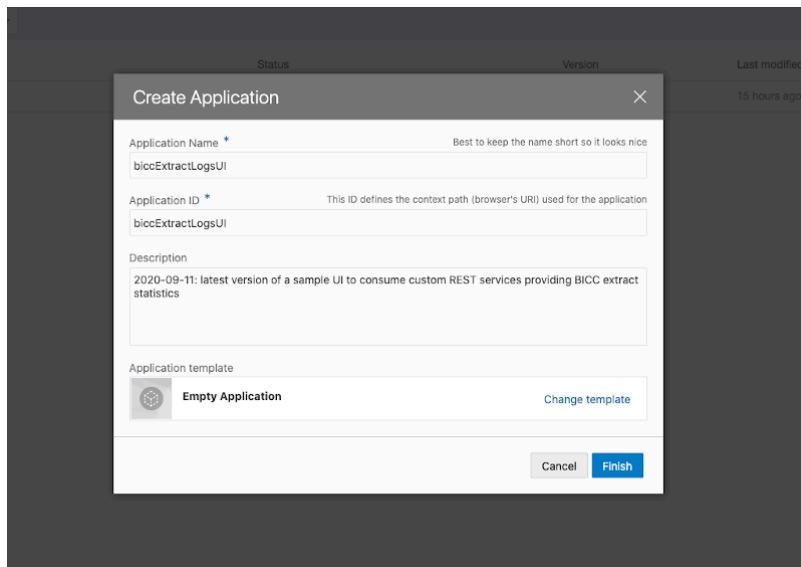
Obviously there are many other ways to get use of these JSON data. One sample will be briefly explained below where we use ***Oracle Visual Builder*** for the further processing of data. Other samples would be invocations from ***Oracle Integration Cloud***, ***Oracle Process Cloud*** or any ***3rd party applications*** that can consume REST payloads for any follow-up tasks.

BICC Log Visualization via Visual Builder

All the previous configurations are culminating now into a situation where we have a Web Server up and running now that delivers BICC log information via a set of REST API's. As just one sample for an UI integration I'd like to highlight the potential usage in Oracle Visual Builder. I did avoid in my sample code the wrapping by a security layer for authentication and session management. Not that this is not important. Just the opposite is true! But the intention was to keep this architecture blueprint as simple as possible - even more in an end to end flow that includes so many components.

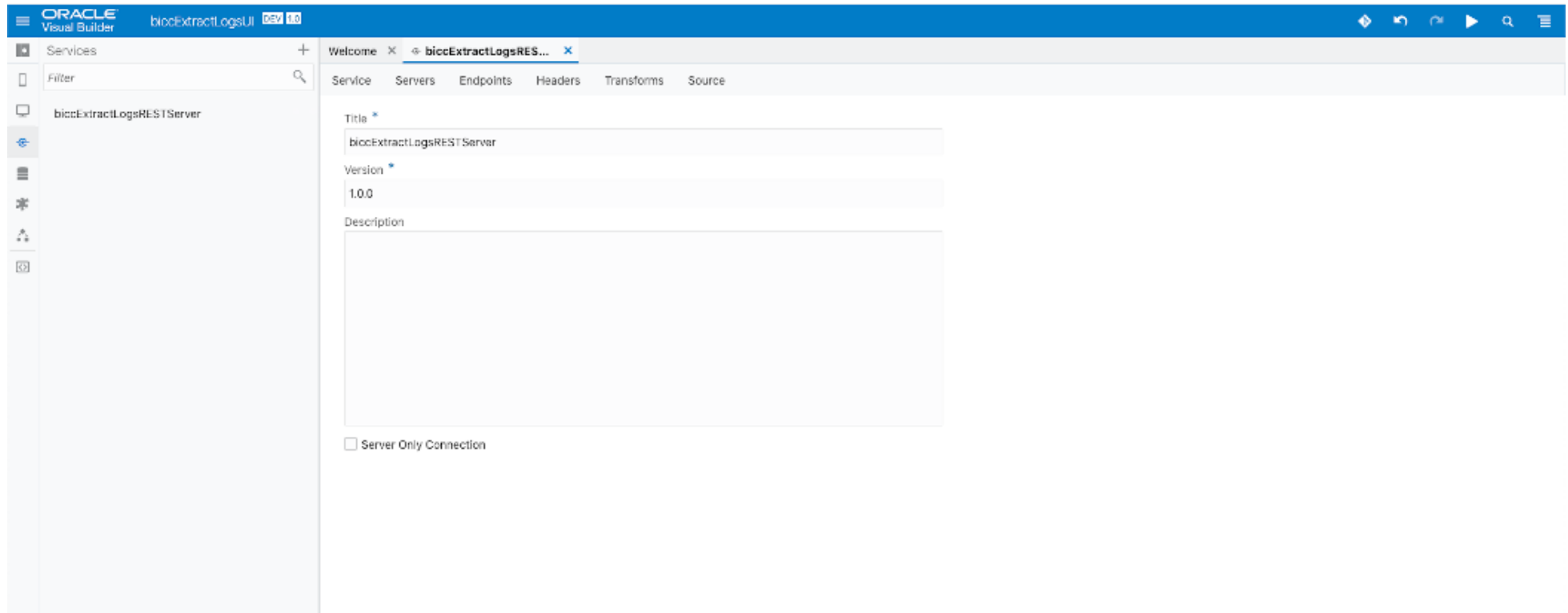
For our VB sample it means we can register these custom BICC log services without any security config. In a real-live or production environment this is obviously not recommended. Also with respect to this quite complex blog I won't provide a full-blown UI application. Its intended to provide just some basic ideas how the creation of such an application can be initiated. I'm definitely not a talented UI designer and so I'll leave this a creative lesson for everyone who is ambitioned to get it done.

As the screenshot below demonstrates we're creating a web application in VB that will use out REST services.



Sample VB App Creation

As shown in screenshot below we can register in VB the Web Server we're running to provide the REST services. We've to use the Public IP Address of our Linux VM including the port for this service.



Sample VB App Service Creation for BICC Log REST Server 1 of 2

The methods we're using to invoke these REST services is GET and we usually retrieve many JSON entries per call. This is a configuration we have to enter when creating a ***Service Connection*** as shown in screenshot below.

Create Service Connection

Method ⁺ URL ⁺

GET http://[redacted]

Action Hint
Get Many ⁺

Start by choosing the HTTP method and giving us the URL for your endpoint. It can optionally include query parameters.

To add path parameters include them in curly braces as shown in the example.

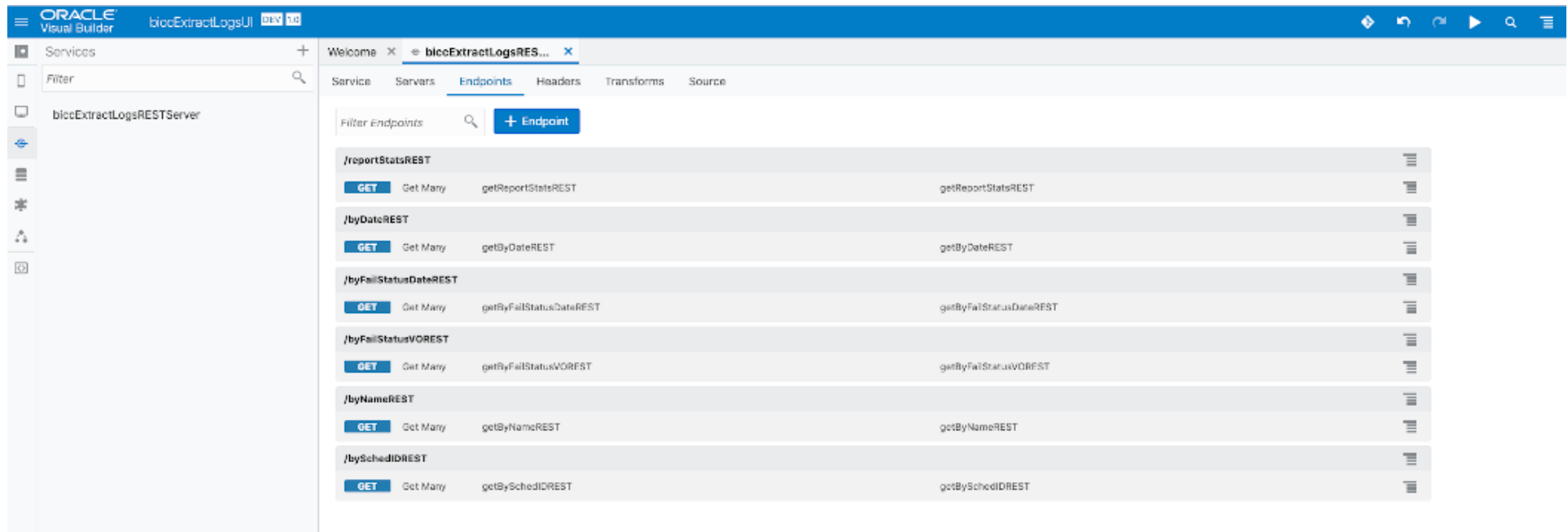
We can help you build apps faster if you select a basic action type for this endpoint (Get Many, Get One, Create, Delete, Update or Custom).

< Back

Cancel Next >

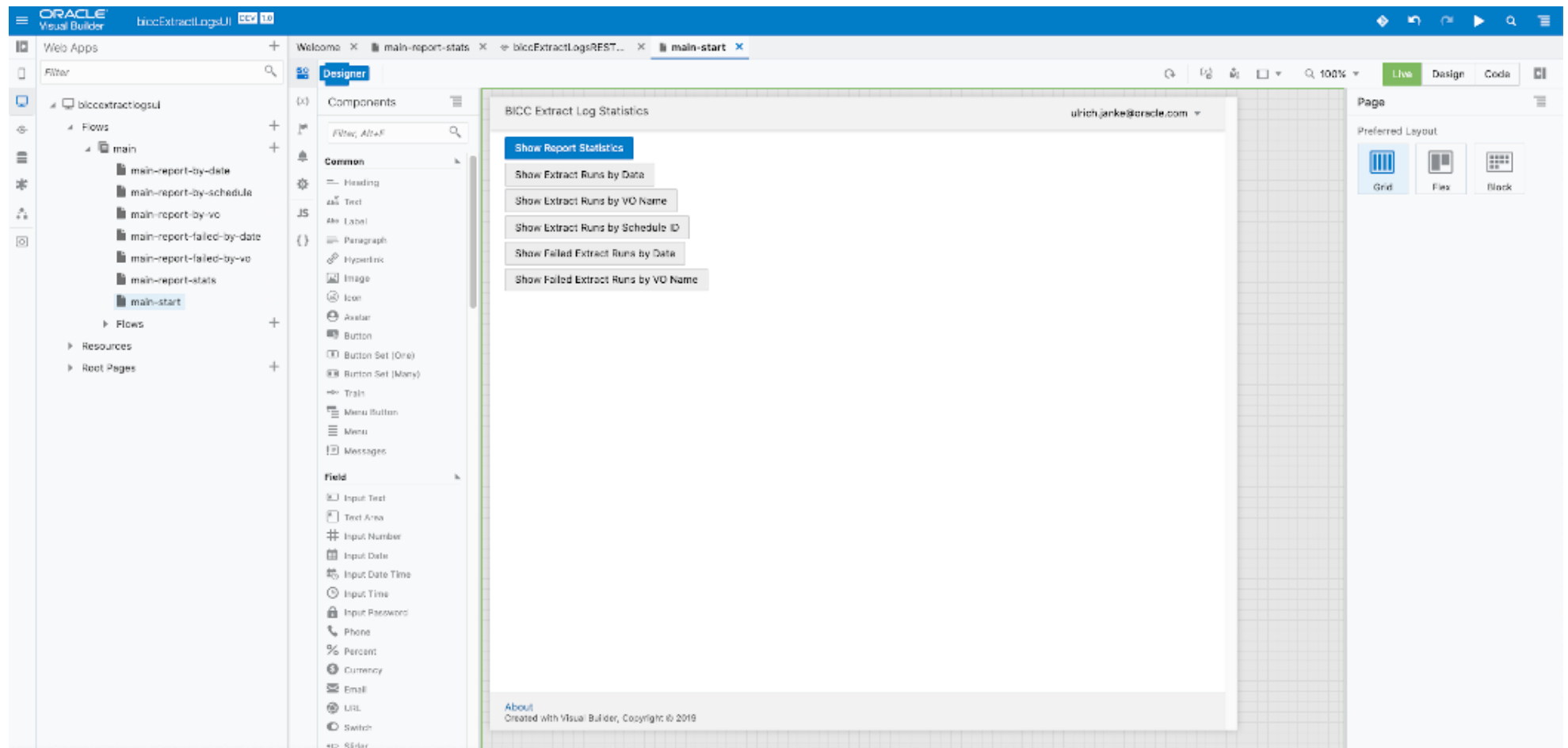
Sample VB App Service Creation for BICC Log REST Server 2 of 2

Once the Service Connection has been created we can register REST Endpoints for that service. The screenshot below shows how these endpoints will appear in VB once we finished that task. From now on these services can be used for UI layouting. There is a broad variety of options to make a use of these services. In case of a significant interest we will consider a dedicated blog highlighting the steps to implement these features.



Sample VB App Service List

I've mentioned above that the UI design is an important, but also quite highly specialized or even a complex task. The last screenshot in this article shows a hint how to use a set of buttons that could control the appearance of the specific views. As mentioned: there are obviously plenty of options to create fancy UI's and as such, those activities should be seen as an activity worth to be handled in an extra blog.



Sample VB App UI Page Design

Conclusion

The purpose of this article was to propose an architecture that uses an OCI native approach to consume extraction output files and to process them. The focus had been set on two main topics: 1) exposing an end-to-end architecture including all participating components. 2) to provide a sample application running on Node.js and Koa.js that processes these log data and runs a set of REST API's. Once such an architecture is in place we can consider other solutions that could make a use of the enhanced capabilities when involving Cloud Native services.