

Unlocking Horizon's secret lair:

A VLSI API Adventure

Tony Foster – Independent Engineering Technologist

Omnissa Tech Insider

September 28, 2026

Disclaimer

Info in this presentation may not be incorporated into any contract

- This presentation may include product features or functionality currently in development.
- Nothing presented constitutes a commitment by Omnissa to deliver any specific product or feature.
- Features, functionality, and timelines are subject to change without notice.
- Do not rely on this information in contracts, purchase orders, or purchasing decisions.
- Final delivery is subject to technical feasibility and market demand.
- Pricing and packaging for any future offerings have not been determined.
- This content is confidential and proprietary to Omnissa.
- It may not be copied, shared, or distributed without prior written consent.
- Certain statements may be forward-looking and are subject to risks and uncertainties.
- All trademarks and product names are the property of their respective owners

About the Wondernerd

- Virtualized 20 years
- VDI/EUC space – 18 years
- Architected / deployed EUC at government agencies and enterprises

- Omnissa Tech Insider
- Connect: linkedin.com/in/wondernerd
- Session content: Wondernerd.net



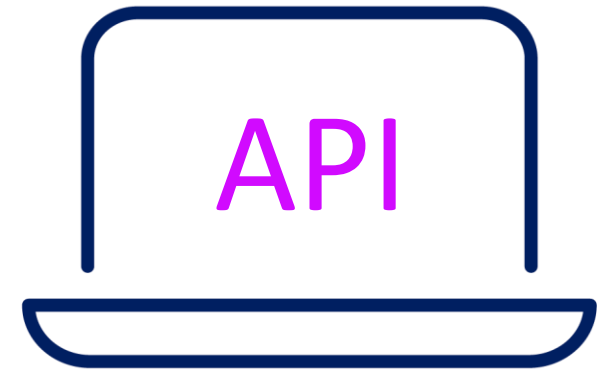
Agenda

- What is the VLSI
- Got Swagger?
- Accessing the API – PowerShell
- Basic structure
- Operations

What is the Virtual Language System Interface (VLSI) API

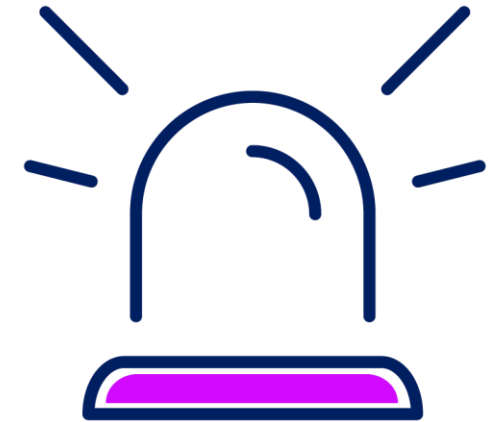
<https://server-address/view-vlsi/sdk>

- API for accessing Horizon View using SOAP
- Backend for Horizon PowerCLI
- <https://developer.omnissa.com/horizon-apis/view/>
- Other Horizon APIs include:
 - Horizon Server
 - Horizon Cloud Service Next Gen
 - Horizon DaaS
 - App Volumes
 - UAG
 - Access
 - Intelligence
 - Identity Service

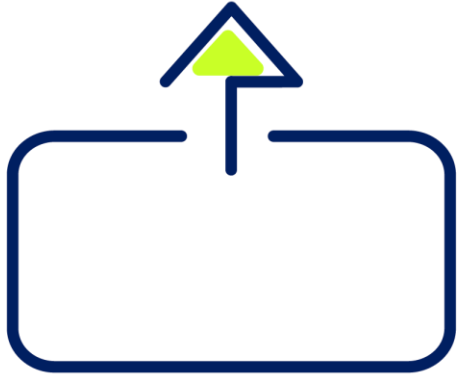


OmniSSA VLSI

- Horizon PowerCLI (SOAP Based)
 - https://kb.omnissa.com/s/article/6000139?lang=en_US
 - ...Deprecation of the Horizon View APIs in Q2 Release of 2026 (earlier planned for Q4 Release of 2025) and its removal in upcoming future releases. Our deprecation of **View APIs** will be handled as follows:
Horizon Q2 2026 Release will be the last version to include the SOAP-based Horizon View APIs. The newer releases after Q2 2026 will only contain the Horizon REST APIs.
- Translation: Horizon 2512 is the last version to have the SOAP based APIs

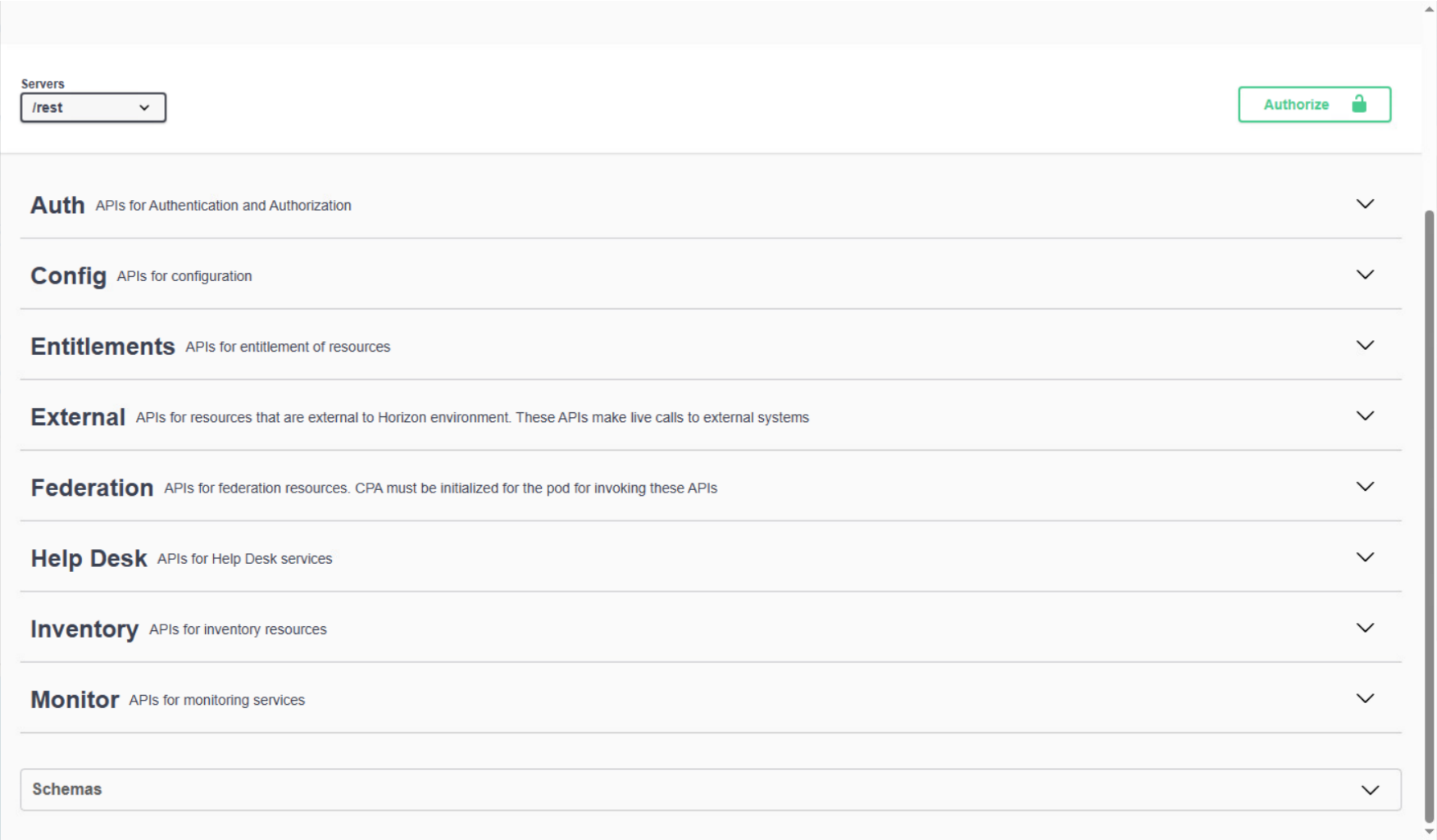


Omnissa Horizon Server API



- Horizon REST APIs
 - <https://developer.omnissa.com/horizon-apis/horizon-server/>
 - Swagger documentation
- Built into Horizon
 - REST Access
 - <https://<connection server>/rest>
 - Swagger
 - <https://<connection server>/rest/swagger-ui/index.html>

Get your Swagger on – top level



Get your Swagger on – action level

The screenshot displays a Swagger API interface with the following endpoints:

- GET** `/inventory/v1/cp-desktop-pools` Lists the capacity provider Desktop Pools in the environment.
- POST** `/inventory/v1/cp-desktop-pools` Creates a capacity provider desktop pool.
- DELETE** `/inventory/v1/cp-desktop-pools/{id}` Deletes a capacity provider desktop pool.
- GET** `/inventory/v1/cp-desktop-pools/{id}` Gets the capacity provider Desktop Pool information.
- PUT** `/inventory/v1/cp-desktop-pools/{id}` Updates capacity provider desktop pool.
- POST** `/inventory/v1/cp-desktop-pools/{id}/action/cancel-scheduled-push-image` Request the cancellation of the current scheduled push image operation or cancellation of promoted secondary image on the specified instant/linked clone desktop pool.
- POST** `/inventory/v1/cp-desktop-pools/{id}/action/promote-pending-image` Promotes pending image to current image in the desktop pool.
- POST** `/inventory/v1/cp-desktop-pools/{id}/action/schedule-push-image` Schedule/reschedule a request to update the image in an automated desktop pool.
- GET** `/inventory/v1/cp-farms` Lists the Farms in the environment.

For the `/inventory/v1/cp-farms` endpoint, the following details are shown:

All the listed privileges are required -

Privileges	Description
POOL_VIEW	Returns results only for those farms, for which the caller has permission to this privilege, on the associated access group of the farm.

Parameters

Name	Description
page integer	page, if passed should be > 0.

A "Try it out" button is located to the right of the parameters section.

Get your Swagger on – action details

POST /inventory/v1/cp-desktop-paste/{id}/actions/promote-guesting-image

POST /inventory/v1/cp-desktop-paste/{id}/actions/schedule-patch-image

GET /inventory/v1/cp-farms

All the listed privileges are required -

Privilege	Description
PODS, VMs	Returns results only for those items, for which the caller has permissions to this privilege, on the associated resource group of the farm.

Parameters

page

Integer

(query)

page, if passed should be > 0.

Example: 1

1

size

Integer

(query)

size, if passed should be > 0.

Example: 10

10

filter

string

(query)

filter expression built using fields with 'supported filters' as described in output model schema of this API.

Example: OrderedMap ["type": "And", "items": List [OrderedMap ["type": "Equals", "name": "cpu", "value": "x86"]]]

["type": "And", "items": [["type": "Eq"

sort_by

string

(query)

Field that would be used to sort the results.

Example: <property>name

<property>name

order_by

string

(query)

Determines the ordering of results.

Available values: ASC, DESC

Default value: ASC

Example: ASC

ASC

Responses

Code	Description	Links
200	OK	API Docs
401	User is not authenticated	API Docs
403	Access to the resource is forbidden	API Docs
429	Too many requests	API Docs

POST /inventory/v1/cp-farms

DELETE /inventory/v1/cp-farms/{id}

Login to API

PowerShell

```
# Bypass cert validation (PS 5.1 & 7 compatible fallback)
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {
$true }
$BaseUrl = "https://[connection server]/rest"
# Construct credentials body for initial login
$Body = @{
    "username" = "api_admin"
    "password" = "SecretPassword"
    "domain"   = "YOURDOMAIN"} | ConvertTo-Json
# Request the login token
$LoginResponse = Invoke-RestMethod -Uri "$BaseUrl/login" -Method Post -
Body $Body -ContentType "application/json"
# Capture the Bearer token (Horizon returns access_token)
$BearerToken = $LoginResponse.access_token
# Define standard headers for future calls
$headers = @{
    "Authorization" = "Bearer $BearerToken"
    "Accept"        = "application/json"}
```

Login to API

PowerShell output – \$Headers

Name	Value
-----	-----
Accept	application/json
Authorization	Bearer eyJhbGciOiJSUzI4NCJ9.eyJ1c2VyLXNpZCI6I1MtMS01LTlxLTI4MDQxNjAwODktMTQzMzAyMjkyNTc5NTM2MjEwLTUwMCIsImNsb3VkaW4iOiJhbnRlLCJ0b2t1biB0eXB1IjoieYWNjZXRzIH...

Basic structure

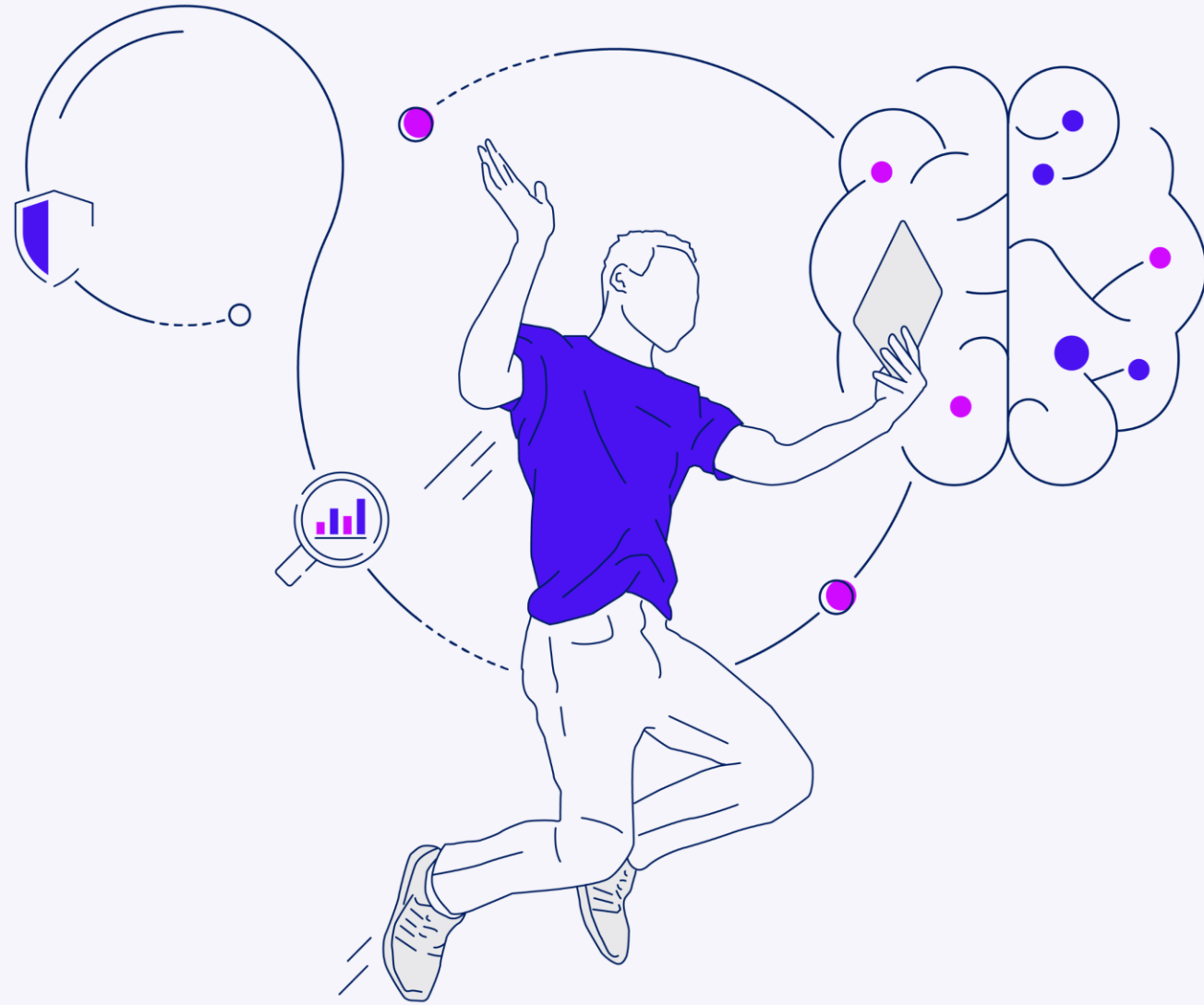


Basic building block

Invoke-RestMethod

```
$BaseUrl = "https://[connection server]/rest"
$APIpath = "[area]/v1/[operation]"
$response = Invoke-RestMethod -Uri
"$BaseUrl/$APIpath" -Method Put -Headers $Headers -Body
$JSONpayload -ContentType Get application/json"
Post
Delete
```

Operations



List desktop pools

Start simple

```
#API location
```

```
$DSPoolPath = "inventory/v1/desktop-pools"
```

```
#API Get call
```

```
$DSPoolList = Invoke-RestMethod -Uri "$BaseUrl/$DSPoolPath" -  
    Method Get -Headers $Headers
```

```
#Display the result
```

```
$DSPoolList
```

List desktop pools

Result

```
id          : 0ec38bc2-fe6a-43b9-996f-17dde7f66a91
name        : API-TEST
display_name : API TESTING POOL
type        : AUTOMATED
source      : VIRTUAL_CENTER
enabled     : True
settings    : @{delete_in_progress=False;
               enable_client_restrictions=False;
               allow_multiple_sessions_per_user=False;
               session_type=DESKTOP; cloud_managed=False;
               cloud_assigned=False; session_settings=;
               display_protocol_settings=}
```

Building on this

Disabling a pool

#Get pool information

```
$DSPoolList = Invoke-RestMethod -Uri "$BaseUrl/$DSPoolPath" -Method Get -Headers  
$Headers
```

#Create JSON list of desktop pools

```
$DSPoolID = $DSPoolList.id
```

```
$DSPoolRawIDs = @("$DSPoolID")
```

```
$DSPoolJSONIDs = ConvertTo-json -inputobject $DSPoolRawIDs #for PowerShell 5.1+
```

#Disable the desktop pool

```
$DSPoolDisResult = Invoke-RestMethod -Uri "$BaseUrl/$DSPoolPath/action/disable"  
-Method Post -Headers $Headers -Body $DSPoolJSONIDs  
-contentType "application/json"
```

Disable desktop pools

Result

id	status_code
--	-----
0ec38bc2-fe6a-43b9-996f-17dde7f66a91	200

Going further

Modifying a pool – gathering the bits

```
#Retrieve Access Group
```

```
$AccessGroups = Invoke-RestMethod -Uri "$BaseUrl/config/v1/local-access-groups"  
-Method Get -Headers $Headers
```

```
#Retrieve pool ID
```

```
$DSPoolList = Invoke-RestMethod -Uri "$BaseUrl/$DSPoolPath"  
-Method Get -Headers $Headers
```

```
$DSPoolID = $DSPoolList.id
```

```
$UpdatedDSPoolPayload = @{  
    "access_group_id" = $AccessGroups.id  
    "allow_multiple_user_assignments" = $true  
    "display_name" = "API TESTING POOL"  
    "enabled" = $false  
    ...}
```

```
#convert payload
```

```
$ConvertedDSPool = $UpdatedDSPoolPayload | ConvertTo-json -depth 5
```

Going further

Modifying a pool – making the call

```
try {  
    $DSPoolUpdate = Invoke-RestMethod -Uri "$BaseUrl/$DSPoolPath/$DSPoolID"  
        -Method Put -Headers $Headers -Body $ConvertedDSPool -ContentType  
        "application/json"  
}  
  
catch [system.net.webexception] {  
    $reader = new-object  
    system.io.streamreader($_.exception.response.getresponsestream())  
    $ResponseText = $reader.readtoend()  
  
    write-error "HTTP Status: $_.exception.response.statuscode"  
    write-error "Server Response: $ResponseText"  
}
```

Going further

Modifying a pool – determining the payload

```
APIcall.ps1 : HTTP Status: The remote server returned an error: (400)
Bad Request..exception.response.statuscode
At line:1 char:1

+ .\OmnissaOneScripts_v4.ps1
+ ~~~~~
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorException
    + FullyQualifiedErrorId :
      Microsoft.PowerShell.Commands.WriteErrorException,OmnissaOneScripts_v4.ps1

C:\Users\Administrator.VDI\Desktop\OmnissaOneScripts_v4.ps1 : Server Response:
{"status":"BAD_REQUEST","timestamp":1789
503512351,"errors":[{"error_key":"inventory.desktop-
pool.provisioning_settings.automated.unset.error","error_message":"
provisioning_settings must be set for automated desktop pools."}]}
```

Going further

Modifying a pool – determining the payload

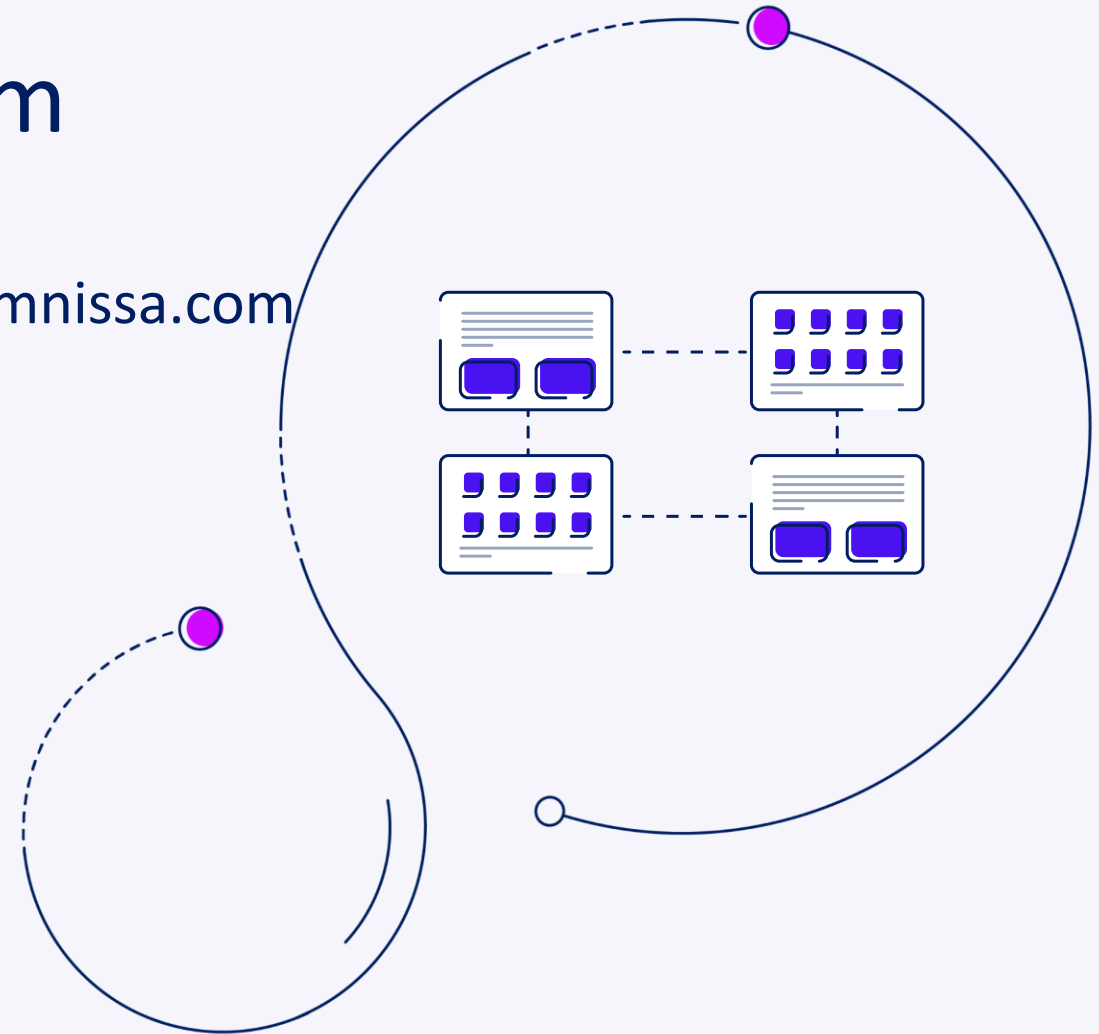
```
503512351, "errors": [{"error_key": "inventory.desktop-  
pool.provisioning_settings.automated.unset.error", "error_message": "  
provisioning_settings must be set for automated desktop pools."}]}
```

Where can we go from here?



Developer.Omnissa.com

- View *code samples* on developer.omnissa.com
- Download shared samples
- Share your code



Thank you

