

HCLSoftware

VoltScript Overview

HCL Labs - Volt MX Go Team



DISCLAIMER

HCL's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at HCL's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

Information regarding potential future products is intended to outline HCL's general product direction and should not be relied upon for any other purpose. Information relating to potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality, all products are made available for sale solely at HCL's discretion and shall be subject to relevant terms and conditions. Development, release, and timing of any future features or functionality described for our products remains at HCL's sole discretion.

Performance is based on measurements and projections using standard HCL benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

Why VoltScript?

Engage 2022

Engage 2023

Early Access 1

Early Access 2

VoltScript in Foundry NOW

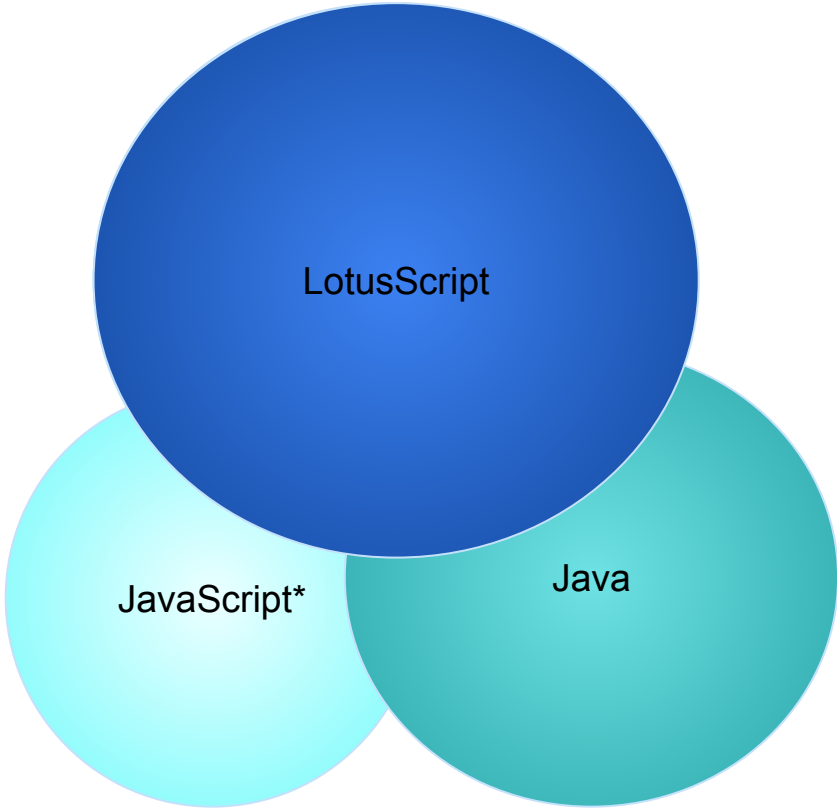
Call To Action

HCLSoftware

Why VoltScript?



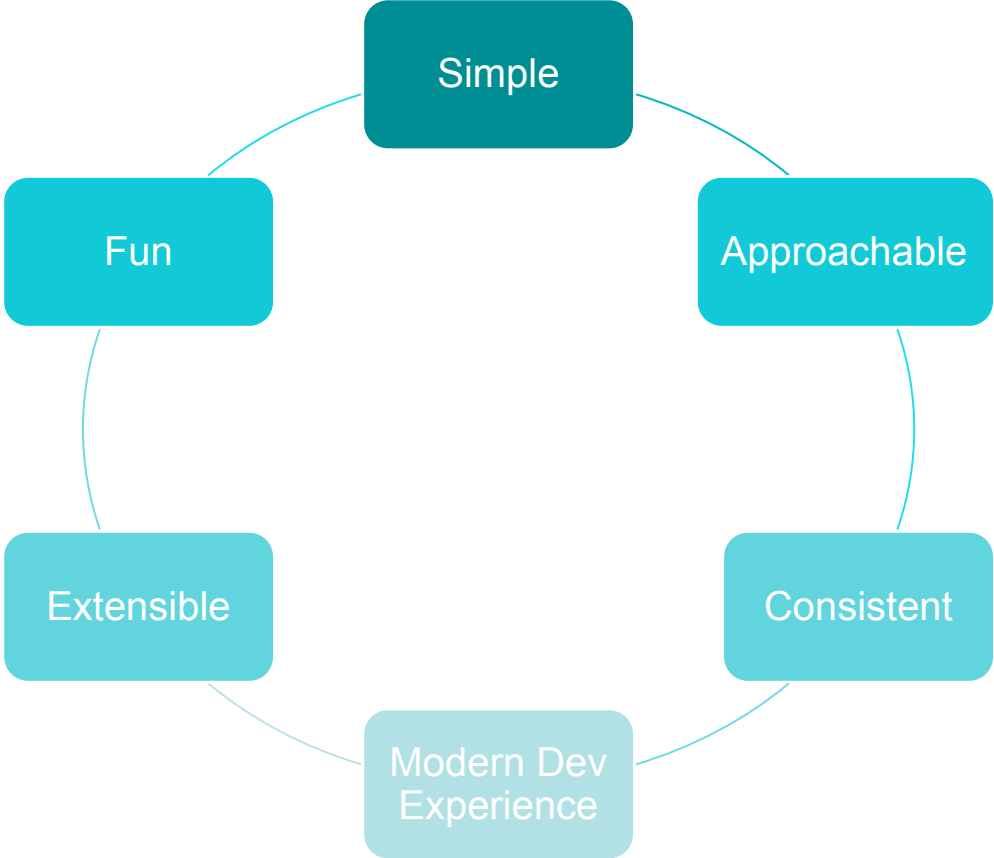
Domino Developer Ecosystem



“Classic” JS
jQuery / Dojo
ECMAScript 5
ECMAScript 6
Node.js
TypeScript

Java 1.5
Java 8
Java 11
Java 17
Java 21

Our Guiding Principles





HCL Volt MX Go

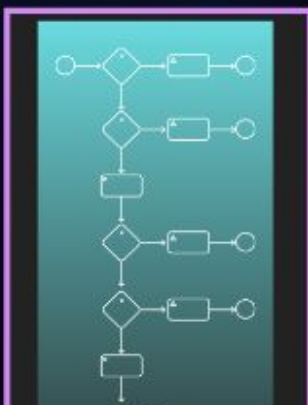
- Web Services
- SaaS Apps
- Databases
- REST APIs
- Domino Danube

VoltScript

- Object Services
- Backend Services
- API Management
- Analytics
- Engagement
- Security



VoltFormula



- Vienna
- London
- El Paso



Apps & Client SDKs

First Touch Setup

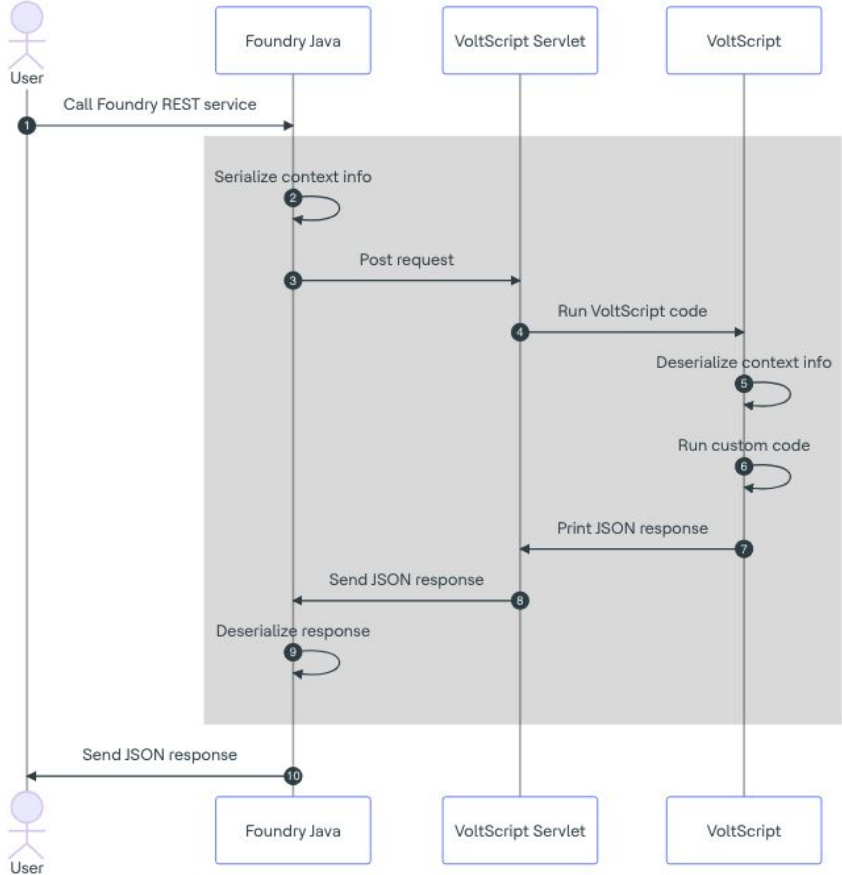
Volt Foundry

Volt Iris IDE

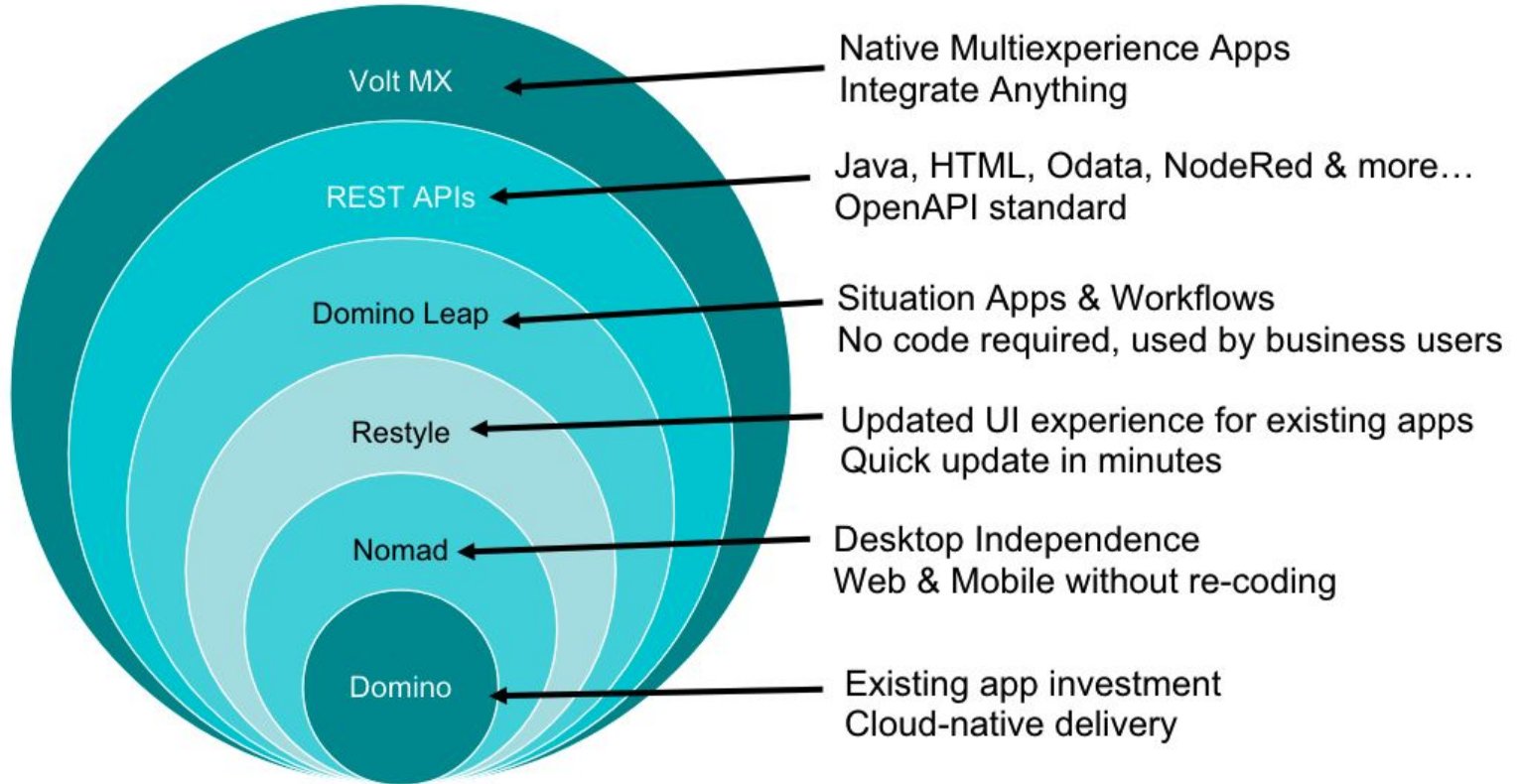
Volt Flare

Design Import

VoltScript is Middleware



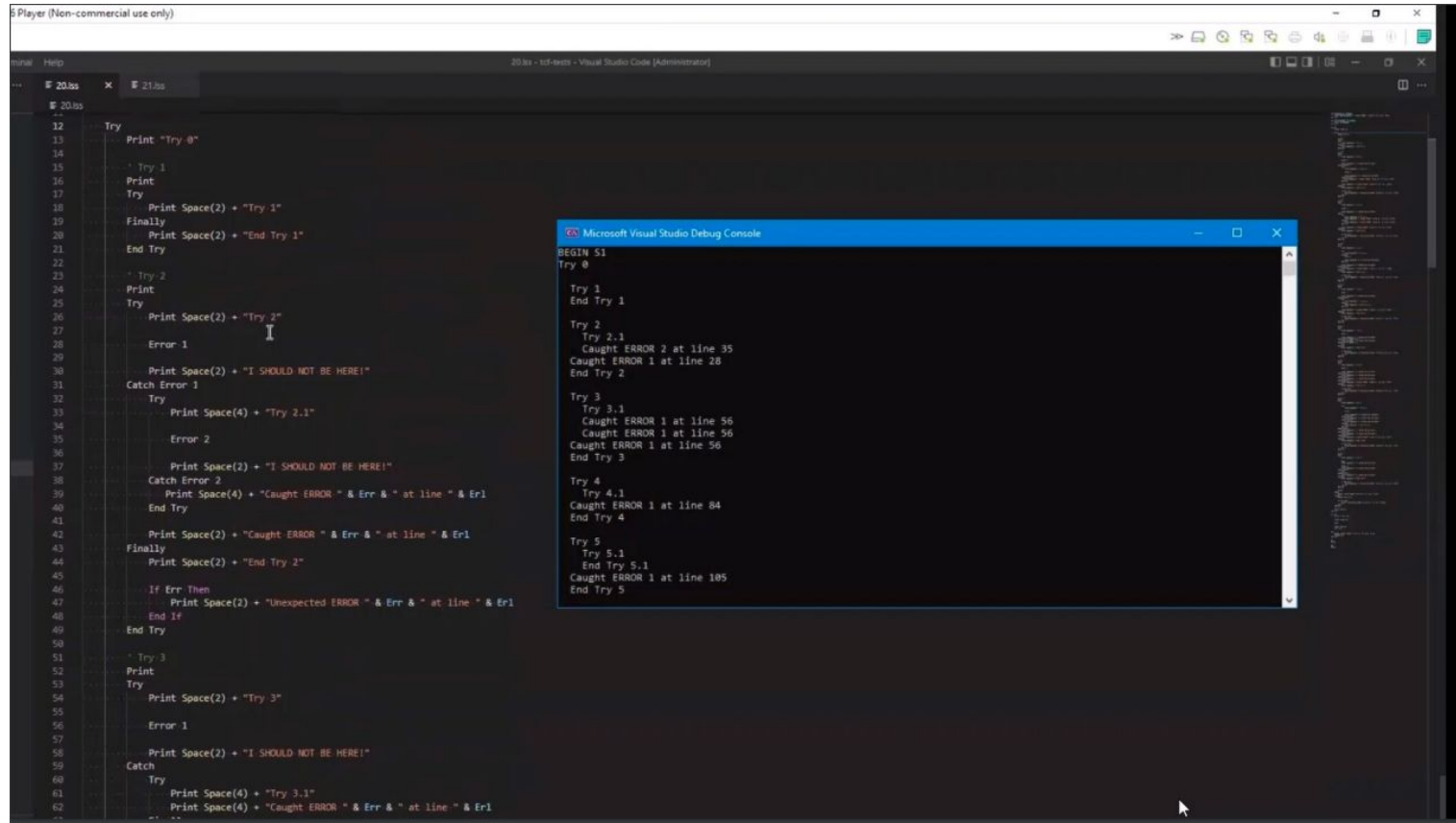
Options for Modernizing your Domino apps



HCLSoftware

Engage 2022

Where were we at Engage 2022?



The image shows a Visual Studio Code window with a C# program and its debug console output. The program uses nested try-catch blocks to demonstrate error handling. The debug console shows the execution flow, including the start of each try block, the execution of its body, and the catching of errors with their line numbers.

```
12 Try
13   Print "Try 0"
14
15   Try 1
16     Print
17     Try
18       Print Space(2) + "Try 1"
19     Finally
20       Print Space(2) + "End Try 1"
21     End Try
22
23   Try 2
24     Print
25     Try
26       Print Space(2) + "Try 2"
27     Error 1
28
29     Print Space(2) + "I SHOULD NOT BE HERE!"
30   Catch Error 1
31     Try
32       Print Space(4) + "Try 2.1"
33
34     Error 2
35
36     Print Space(2) + "I SHOULD NOT BE HERE!"
37   Catch Error 2
38     Print Space(4) + "Caught ERROR " & Err & " at line " & Err
39   End Try
40
41   Print Space(2) + "Caught ERROR " & Err & " at line " & Err
42 Finally
43   Print Space(2) + "End Try 2"
44
45   If Err Then
46     Print Space(2) + "Unexpected ERROR " & Err & " at line " & Err
47   End If
48 End Try
49
50
51 Try 3
52   Print
53   Try
54     Print Space(2) + "Try 3"
55
56   Error 1
57
58   Print Space(2) + "I SHOULD NOT BE HERE!"
59 Catch
60   Try
61     Print Space(4) + "Try 3.1"
62     Print Space(4) + "Caught ERROR " & Err & " at line " & Err
```

```
BEGIN 51
Try 0
  Try 1
  End Try 1
  Try 2
  Try 2.1
  Caught ERROR 2 at line 35
  Caught ERROR 1 at line 28
  End Try 2
  Try 3
  Try 3.1
  Caught ERROR 1 at line 56
  Caught ERROR 1 at line 56
  Caught ERROR 1 at line 56
  End Try 3
  Try 4
  Try 4.1
  Caught ERROR 1 at line 84
  End Try 4
  Try 5
  Try 5.1
  End Try 5.1
  Caught ERROR 1 at line 105
  End Try 5
```

HCLSoftware

Engage 2023

Where were we last year?

Language Enhancements

Supported stack trace

Return keyword

New mathematical operators (++ etc)

New inequality operator alias (!= as well as <>)

Short-circuit conditionals (&& and ||)

Developer Environment

VS Code extension

DevContainer

Code running in VS Code, command line and Java

Foundry pre/postprocessor POC

Unit tests

Dependency Management

VSID (revamped LSX Toolkit)

Where were we last year?

VoltScript Extensions

ContextVSE

StreamVSE

JsonVSE

WebVSE

ZuluVSE

OSUtilsVSE

ZipVSE

HashVSE

KeepVSE

CouchVSE

VoltScript Library Modules

VoltScript Testing Framework

Already open sourced

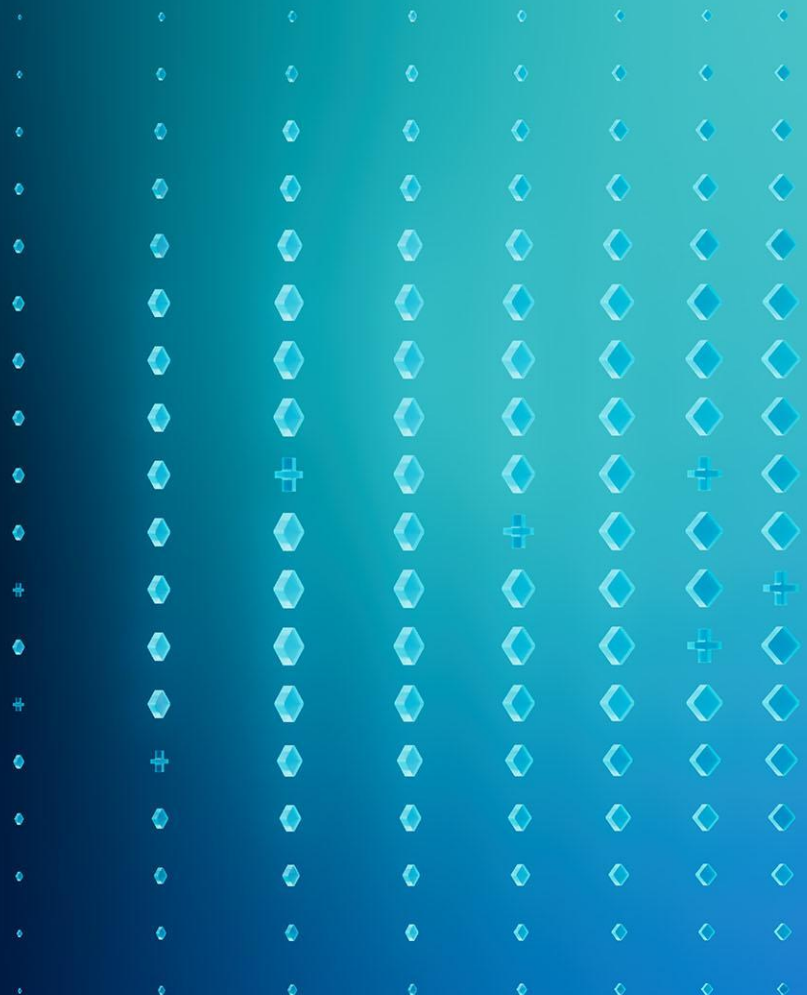
Also ported to LS on OpenNTF

VoltScript JSON Converter

VoltScript Collections

HCLSoftware

Early Access 1



Early Access 1

Released September 2023

Enhanced VSEs

Published to Volt MX Marketplace

Automated build pipelines and testing

VoltScript JSON Converter and VoltScript Collections open sourced

VoltScript Console Colors open sourced

DevContainer Docker image on Harbor

Windows zip file

Build Management VS Code extension

Full documentation

API docs and unit test reports

How-to guide samples and tutorials

HCLSoftware

Early Access 2 / Engage 2024



Early Access 2

VoltScript VoltMX Middleware

Updated VSEs

- Working in Foundry

- New APIs and fixes

XML VSE

New Windows package

- Now on HCL Software downloads site

New Docker image



Hot Off The Press!

Further enhancements to VSEs

You talk, we act

VSEs and VoltScript runtime updated for ubi-8 minimal

Updated dev container - ubi-8 minimal

Fixes to VoltScript JSON Converter and VoltScript Collections

VoltScript Interface Designer open sourced

VoltScript HTTP server - internal access only

VoltScript debug server

VoltScript VoltMX Middleware

New tutorials - get ready for Foundry

VoltScript in Foundry

Integration Services

Unit testing and local HTTP testing context

Edit in Foundry

Pre/postprocessors

Snippets

Monaco editor content assist

Foundry snippets for atlas.json and integration services

Package for Foundry

LS → VS planning

LotusScript → VoltScript

NotesUIDocument, Selected documents → UNIDs as input parameters ← Same as XPages, Domino web

NotesUIWorkspace.Prompt/DialogBox → input parameters ← Same as XPages, Domino web

NotesSession → KeepServer + JWT token

NotesDatabase → KeepScope

NotesDocument → KeepDocument

NotesItems → JSON objects

NotesViewEntryCollection → JSON object

View entry / document collection update → KeepScope bulk operations ← embrace DQL

OpenLog → logging, KeepDocument create

Print statements, messageboxes, refreshing UI → JSON return messages ← Same as XPages

HINT: Profile your agents and LotusScript libraries, identify optimisations for remote running

Call to Action

- Download VoltScript EA2
- Think “testable code” – use LotusScript port of VoltScript Testing
- Think “middleware” – HTTP timeouts, data is remote and JSON
- Read the documentation
- Try the samples
- Review Language Enhancements
- Review VoltScript Extensions and Libraries
- Share your thoughts and experiences

Questions?



Get Started with Volt MX Go today!

Hosted Free Trial
Try it now on HCL Sofy!



Learn

HCL Software U
- Self Learning
Courses



VoltScript
Preview
- Early Access



Webinar Series
- Intro and Deep
dive content



BP Enablement
Webinar



HCLSoftware

[hcltechsw.com](https://www.hcltechsw.com)