**Securely & dynamically manage DB credentials with HashiCorp Vault for “webMethods Adapter for JDBC” connection**

*Overview*: **Vault** is a secret management and data protection tool from a company called HashiCorp. It secures stores and manages a tight control on the access of the secrets.

In this demonstration, it will be shown as to how Vault can be utilized for our webMethods JDBC adapter connections to dynamically generate username & password based on a role defined. This eliminates the case of relying on a static username & password. It introduces a new ability which is with every service/end user accessing the database with unique credentials, it makes auditing much easier when questionable data access is discovered: you can track it down to the specific instance of a service based on the SQL username.

*Tools used*: HashiCorp Vault, cmd.exe, Integration Server, Integration Administrator web UI, wMDesigner.

*Disclaimer*: The cmd.exe will always be opened from directory of vault.exe if the path environment variable is not set. In this demo, cmd.exe will be started from the directory of vault.exe

*Step1*: Download Vault.

Invoke the browser and navigate to <https://www.vaultproject.io/downloads.html> as shown below.



 **Fig.1**

Based on the OS and the processor bit, download the appropriate flavor of Vault in your local or on a cloud machine. The demo here will considering the case that Vault has been downloaded in the local Windows 7 (64bit)

*Step2*: Once the installation is complete, the server needs to be started. In this demo, the server will be started in ‘dev’ mode. ‘Dev’ mode is not the preferred mode for starting the Vault server in production environment as it is not very secure. A real Vault server is usually started in conjunction with ‘Consul’ which is also a tool from HashiCorp.

Open the cmd.exe, and execute the below command

* vault server -dev

The screen should look like this,



 **Fig.2**

The server keeps on running in the foreground and hence the window should not be closed. The ‘dev’ server stores all its data in-memory.

*Step3*: Launch a new cmd.exe and execute the command below,

* set VAULT\_ADDR=http://127.0.0.1:8200(as per the instruction in Fig.2)

This execution helps the vault client to talk to vault server.

Save the Unseal key and the Root Token safely.

Set an environment variable “VAULT\_DEV\_ROOT\_TOKEN\_ID” with the value as the Root Token as shown below at the prompt,

* set VAULT\_DEV\_ROOT\_TOKEN\_ID="s.o3S8j2bGY7FCxWPuBYfCK7qi"

The Unseal key and the Root Token can be found when the server is started properly as shown above in Fig.2 highlighted in red.

*Step4*: Verify the server is running by executing the below command,

* vault status

The screen should look like this,



 **Fig.3**

*Step4*: Create a MS SQL DB using the root username & password

A database named TEST\_DB with table named Person has been created from SQL Server Management Studio.

Below is the screenshot,

 **Fig.4**

*Step5*: The enabling of the database secrets engine and role creation will be done. The MSSQL database secrets engine for Vault generates database credentials dynamically based on configured roles. This means that services that need to access a database no longer need to hardcode credentials: they can request them from Vault, and use Vault's leasing mechanism to roll out credentials.

Enable the database secrets engine, by executing the command as shown below,

* vault secrets enable mssql

The screen below shows the result of execution,



 **Fig.5**

Now, the Vault needs to know, how to connect to an MSSQL instance. This is done by writing the connection string by executing the command as shown below,

* vault write mssql/config/connection ^ connection\_string="server=localhost;port=1433;userid=<rootUser>;password=<YourPassword>;database=TEST\_DB;app name=vault;"



 **Fig.6**

Ignore the warnings to proceed.

We can also configure the lease settings for the Vault by executing the command as shown below,

* vault write mssql/config/lease ^

 ttl=2h ^

 max\_ttl=24h

This forces an application to renew their credentials at least 2 hourly, and to recycle them once per day. The screen should like this,



 **Fig.7**

The next step is to configure a role. A role is a logical name that maps to a policy used to generate those credentials. For example, let’s create an "onlyReadAllowed" role: as shown below,

* vault write mssql/roles/**onlyReadAllowed** ^

 sql="CREATE LOGIN [{{name}}] WITH PASSWORD = '{{password}}'; USE **TEST\_DB**; CREATE USER [{{name}}] FOR LOGIN [{{name}}]; GRANT SELECT ON SCHEMA::dbo TO [{{name}}]"

Above, only SELECT privilege has been granted to the role **onlyReadAllowed**. Any privilege can be granted.

The screen should look like this:



 **Fig.8**

*Step6*: Retrieve DB credentials dynamically. Execute the command as shown below,

* vault read mssql/creds/onlyReadAllowed

The screen should look like this,



 **Fig.9**

If we now refresh the SQL Management Studio, we should see the user created there. Below is the screenshot. 

 **Fig.10**

This user credential will be valid for a period of 2 hours and will be refreshed at the end of 24 hours.

*Step7*: Create a “webMethods Adapter for JDBC” connection as shown below,



 **Fig.11**

Enable the connection after creation as shown in the figure below,



 **Fig.12**

Step8: Create an Adapter Service (SelectStmt) to fetch the values from the Person table in the TEST\_DB using the connection created and enabled above. Once the adapter service is executed, the results are displayed in the figure below,



 **Fig.13**

If the results displayed are matched with the Person table of the TEST\_DB table, we can see the same set of entries as shown below.



 **Fig.14**

Every time a new set of credentials are required, just execute the command as shown below and it will produce a new set of credentials with a new user being created which can be validated from SQL Management Studio,

* vault read mssql/creds/onlyReadAllowed

For any clarification please feel to reach out at avde@softwareag.com