Proposed REST Interface to VistA

Rob Tweed
M/Gateway Developments Ltd

Twitter: @rtweed
Email: rtweed@mgateway.com

http://www.mgateway.com

REST Access to VistA

- "Recipe Book" for REST Access to all of VistA's functionality
- Supported today by EWD.js
- One single classification/pattern of services types
  - Two for login / authentication process
  - One for all other functions thereafter

VistA REST Classification

- initiate
  - Requests a new EWD Session and one-time encryption key
- authenticate
  - Secure REST service for Access Code/Verify Code authentication
- All other functions

initiate

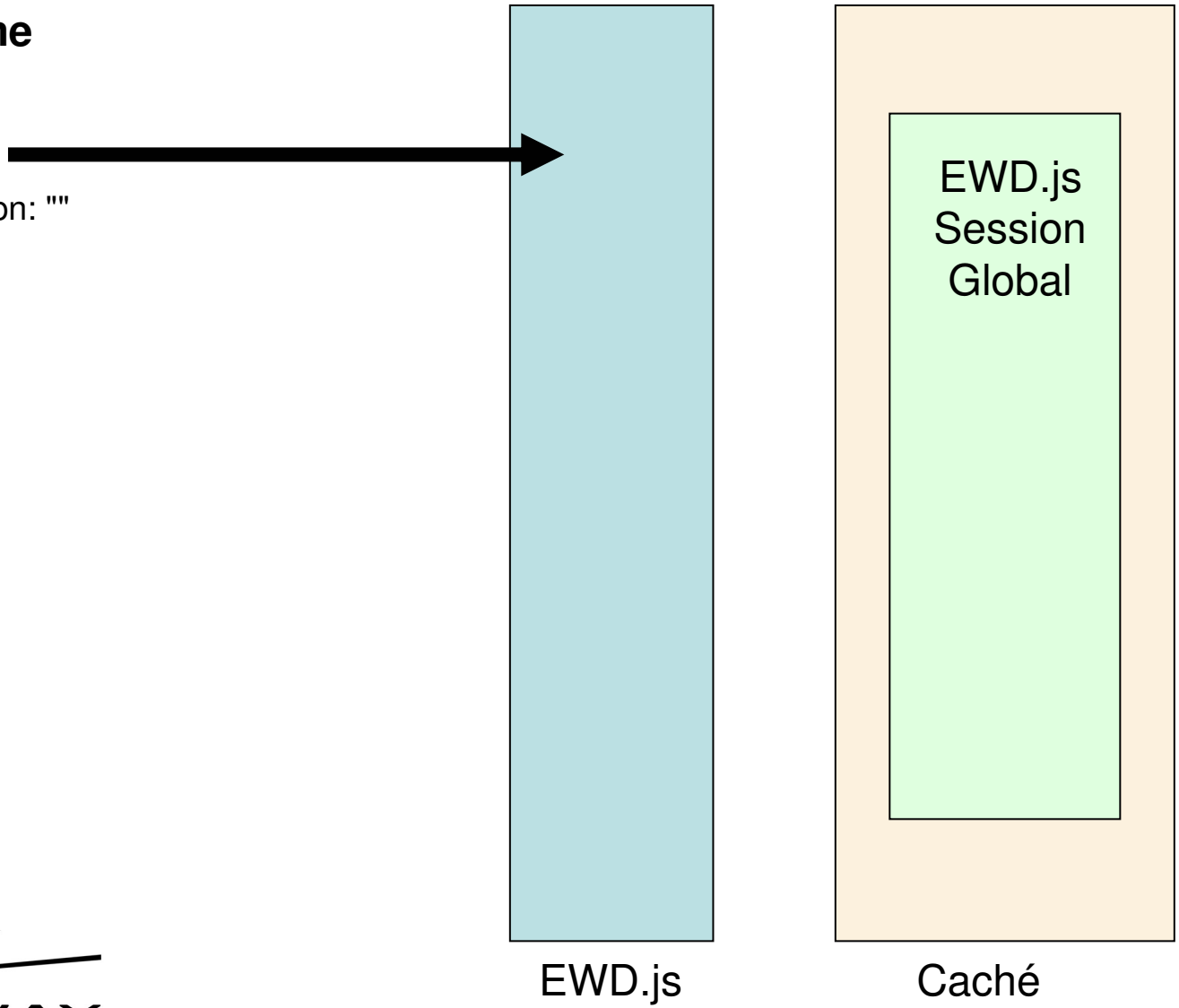
- Starts the authentication sequence
- Unauthenticated request
  - No or empty Authorization Header
- No access to VistA possible without this first step

initiate

Method Name

initiate

Authorization: ""

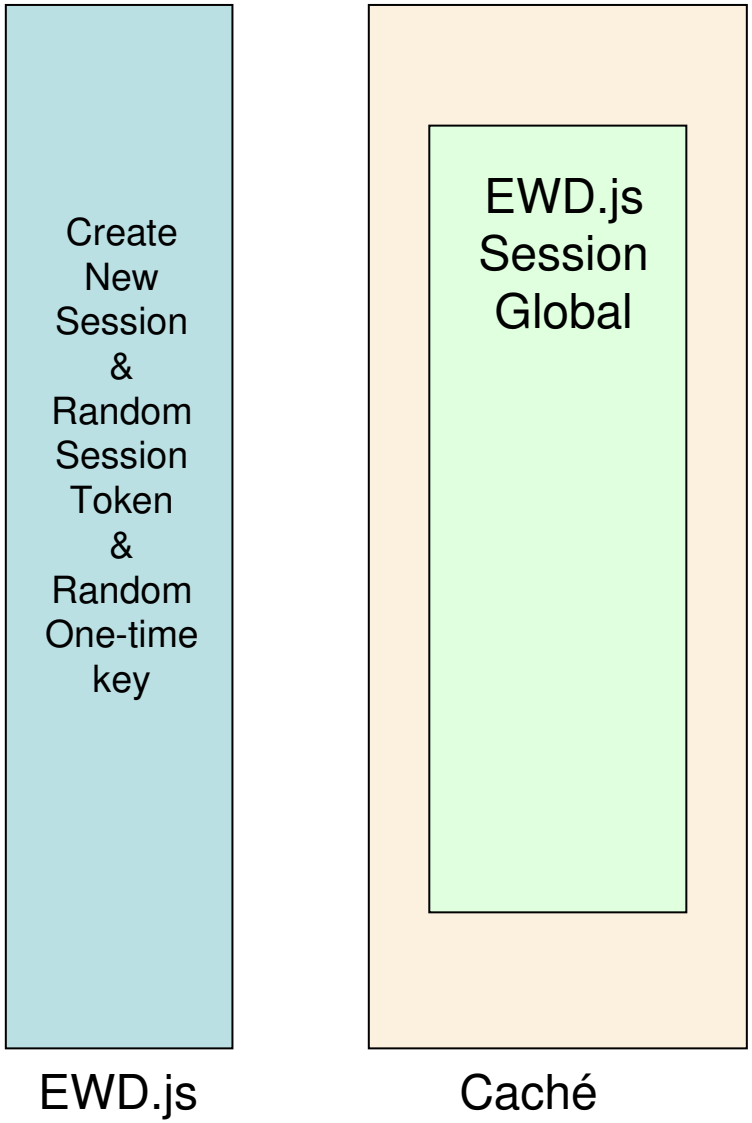


initiate

Method Name

initiate

Authorization: ""

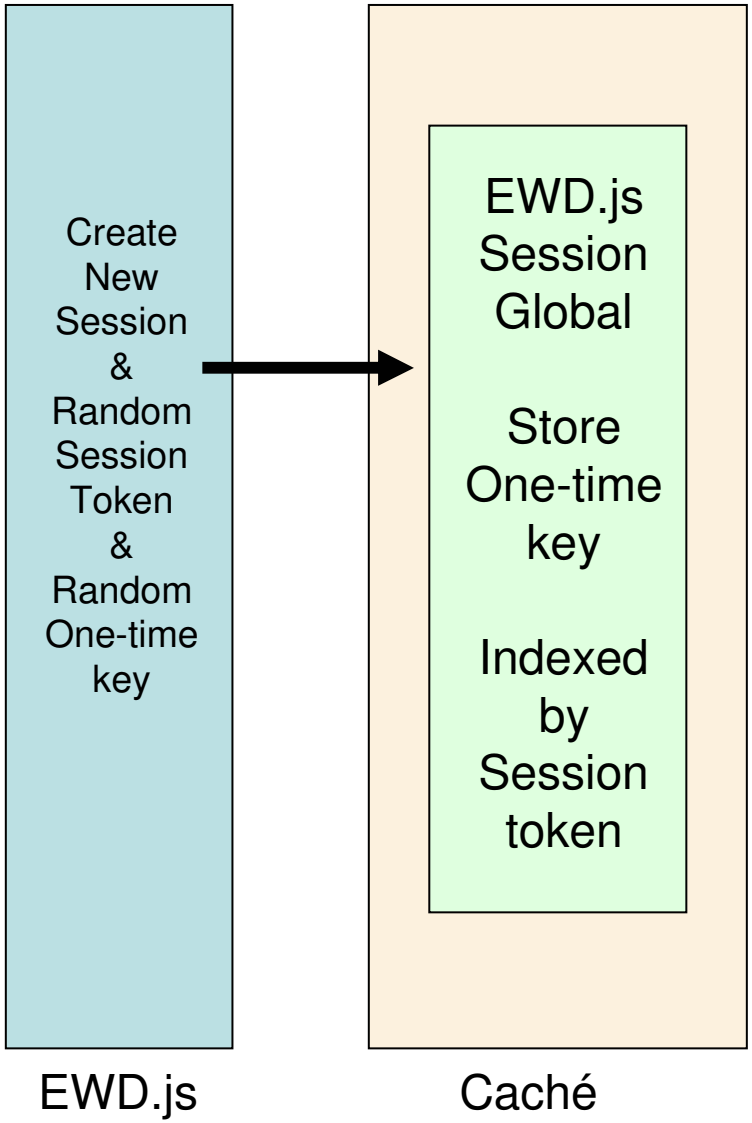


initiate

Method Name

initiate

Authorization: ""



initiate

Method Name

initiate

Authorization: ""

JSON Response

Authorization
(Session)
Token
+
One-time key

Create
New
Session
&
Random
Session
Token
&
Random
One-time
key

EWD.js

EWD.js
Session
Global

Store
One-time
key

Indexed
by
Session
token

Caché

authenticate

- Provides a secure way of sending Access Code & Verify Code via REST to VistA
- One-time key from *initiate* response is used to encrypt the Access Code & Verify Code
- Authorization value from *initiate* response is used as Authorization Header
  - Points EWD.js to EWD Session holding one-time key

Client: Create Encrypted login

1) Construct string using Access Code & Verify Code, eg:

accessCode=fakedoc1&verifyCode=ABC123((

2) : Encrypt this string using the one-time key and aes-256-cbc algorithm:

qwe6876qw78097ewfjsdflkfsdlf90890wewekjdsfsd

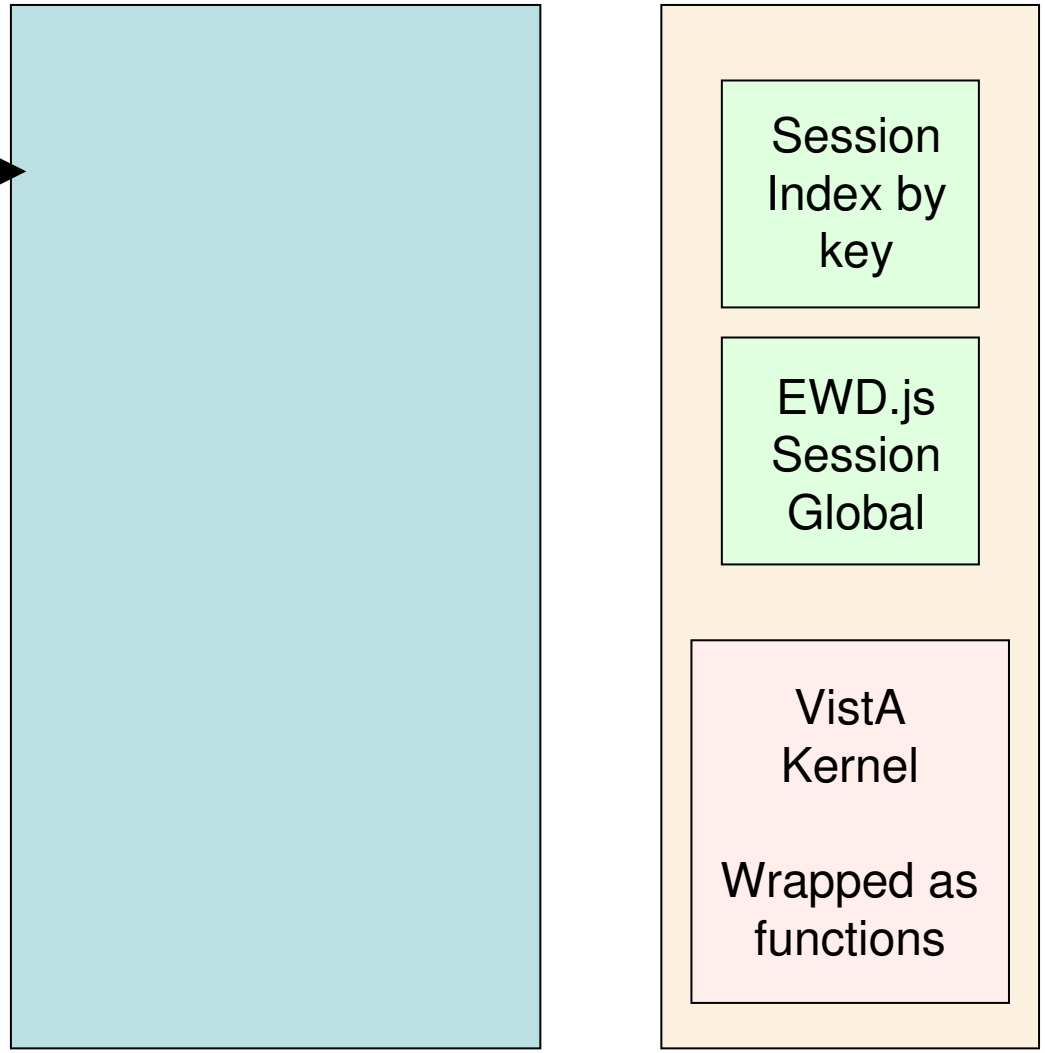
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



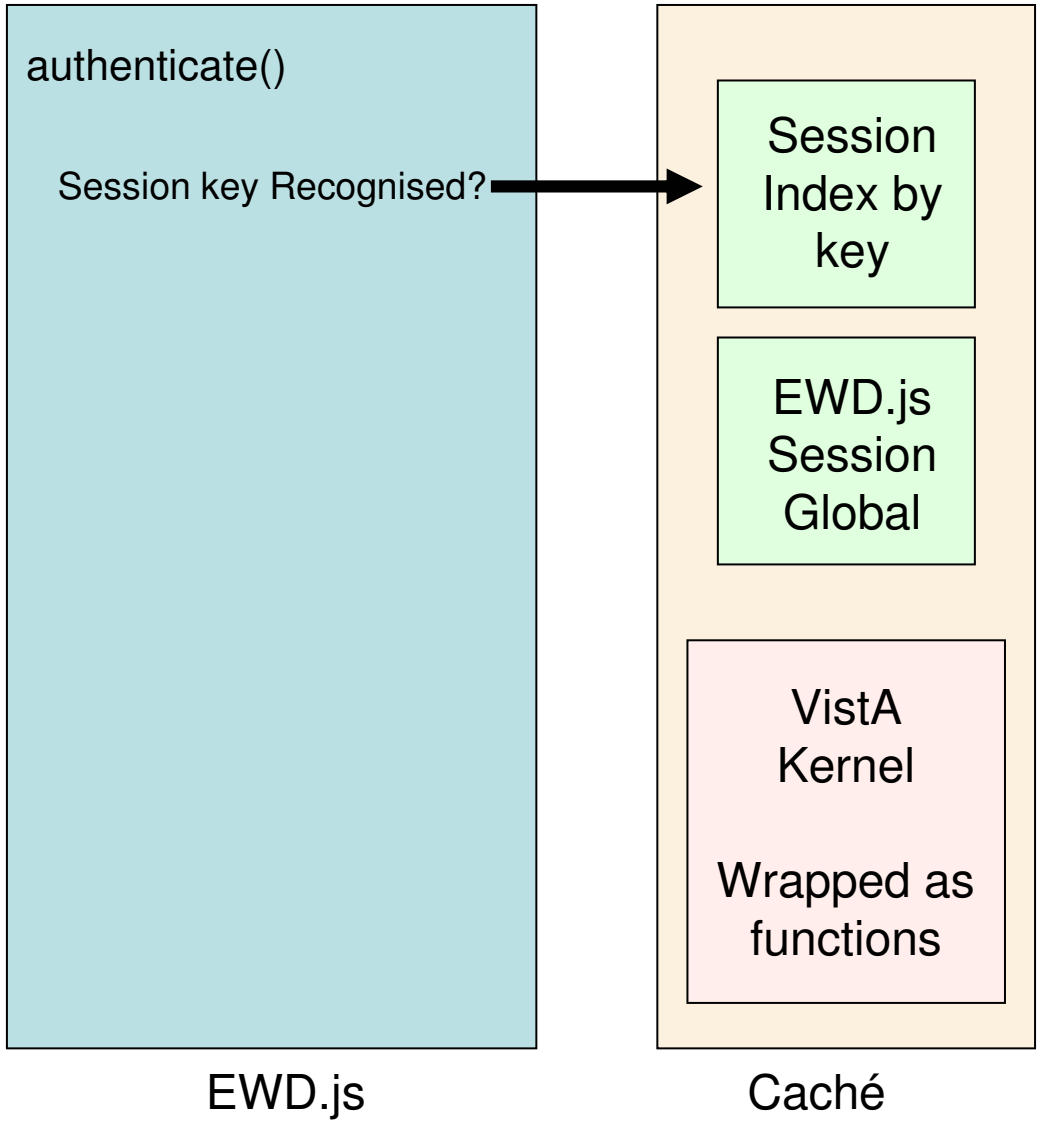
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



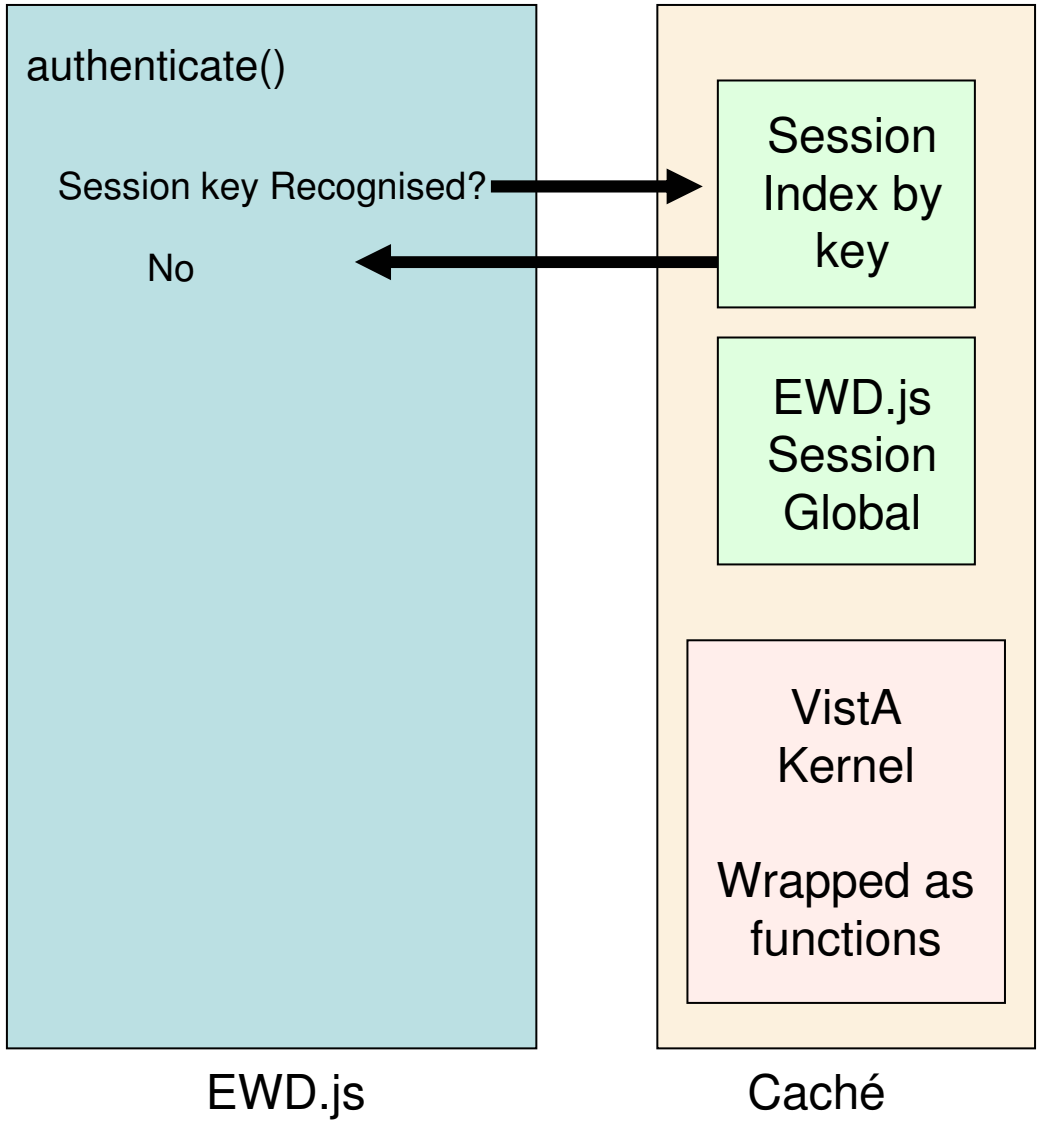
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



authenticate

Method Name

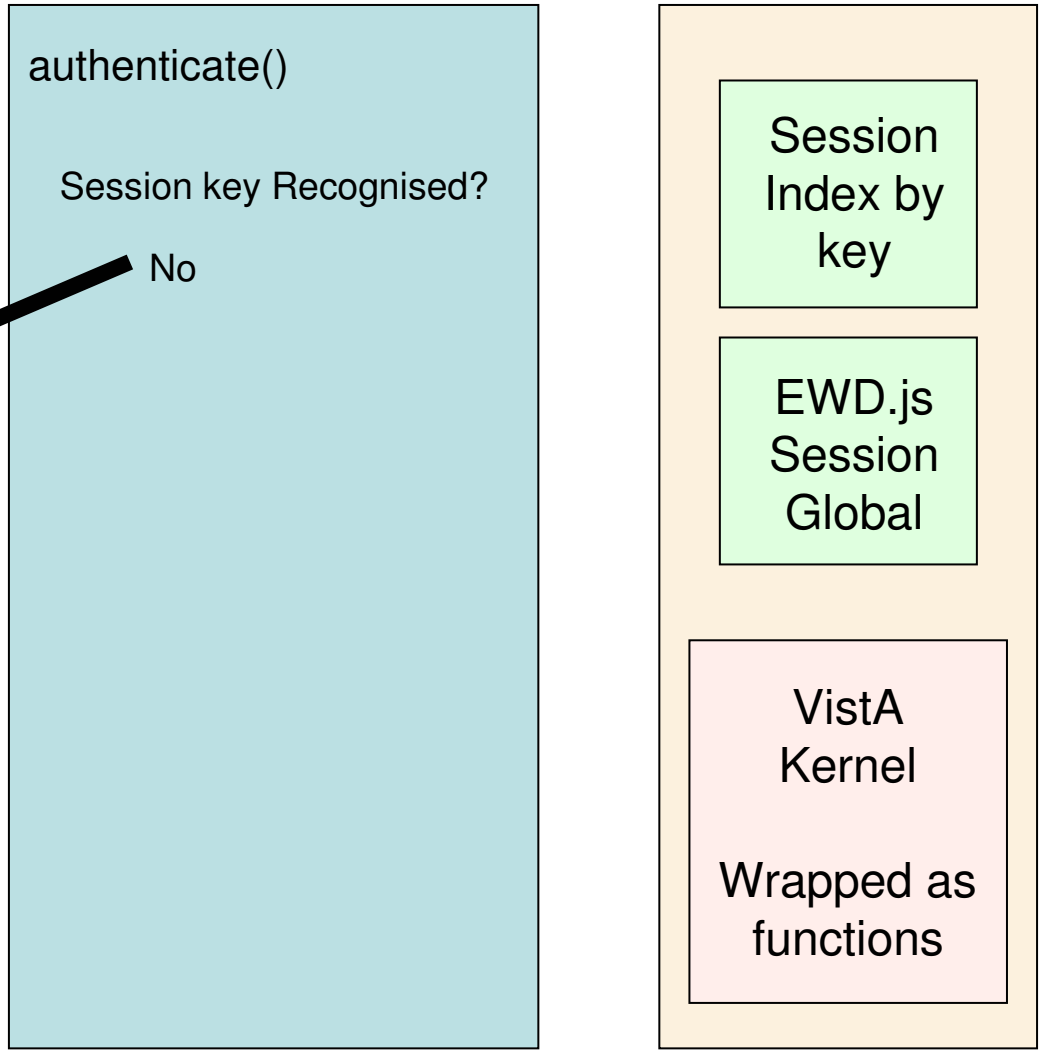
authenticate

credentials=: encrypted string

Authorization: session key

JSON Response

401 Error



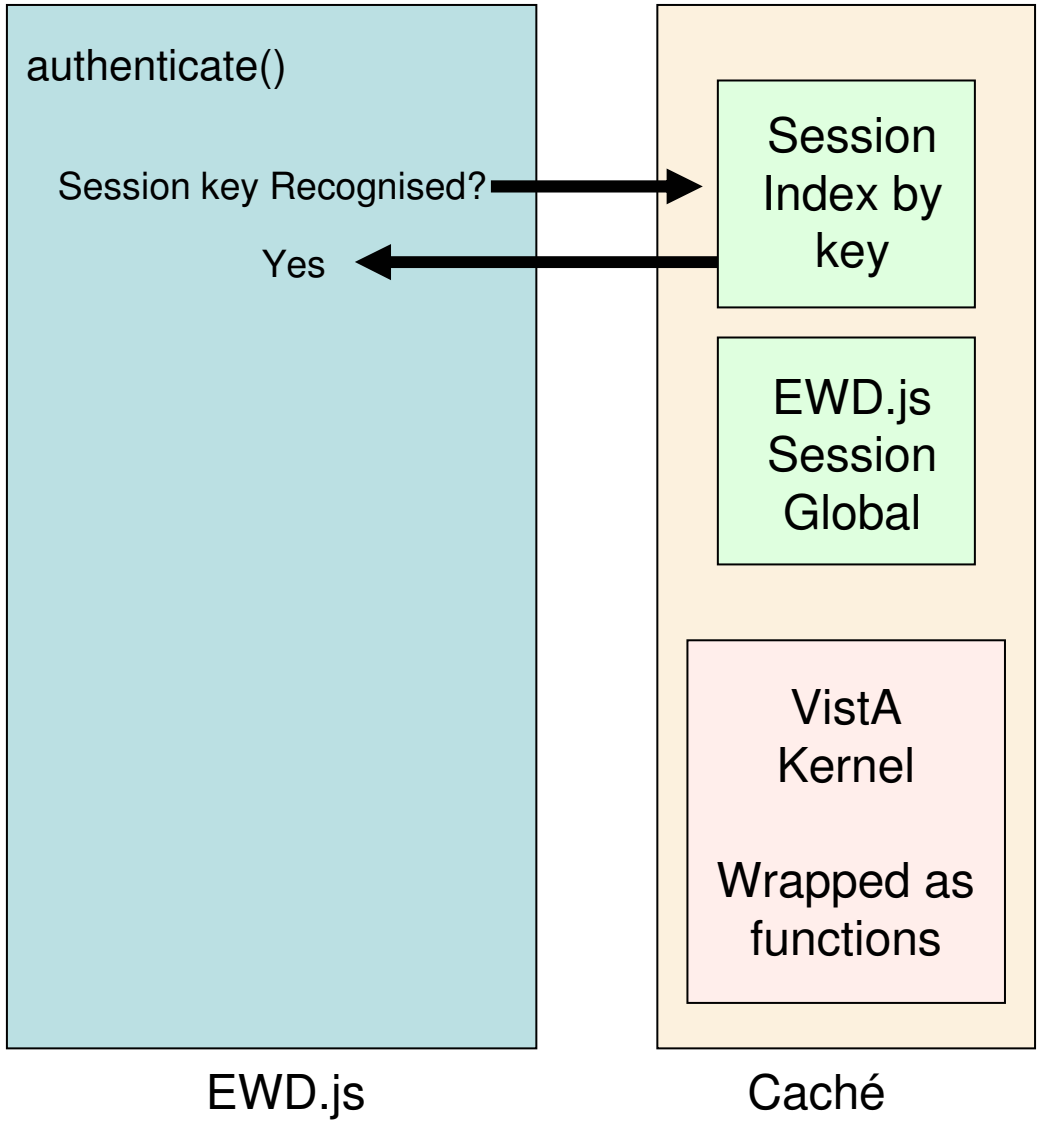
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



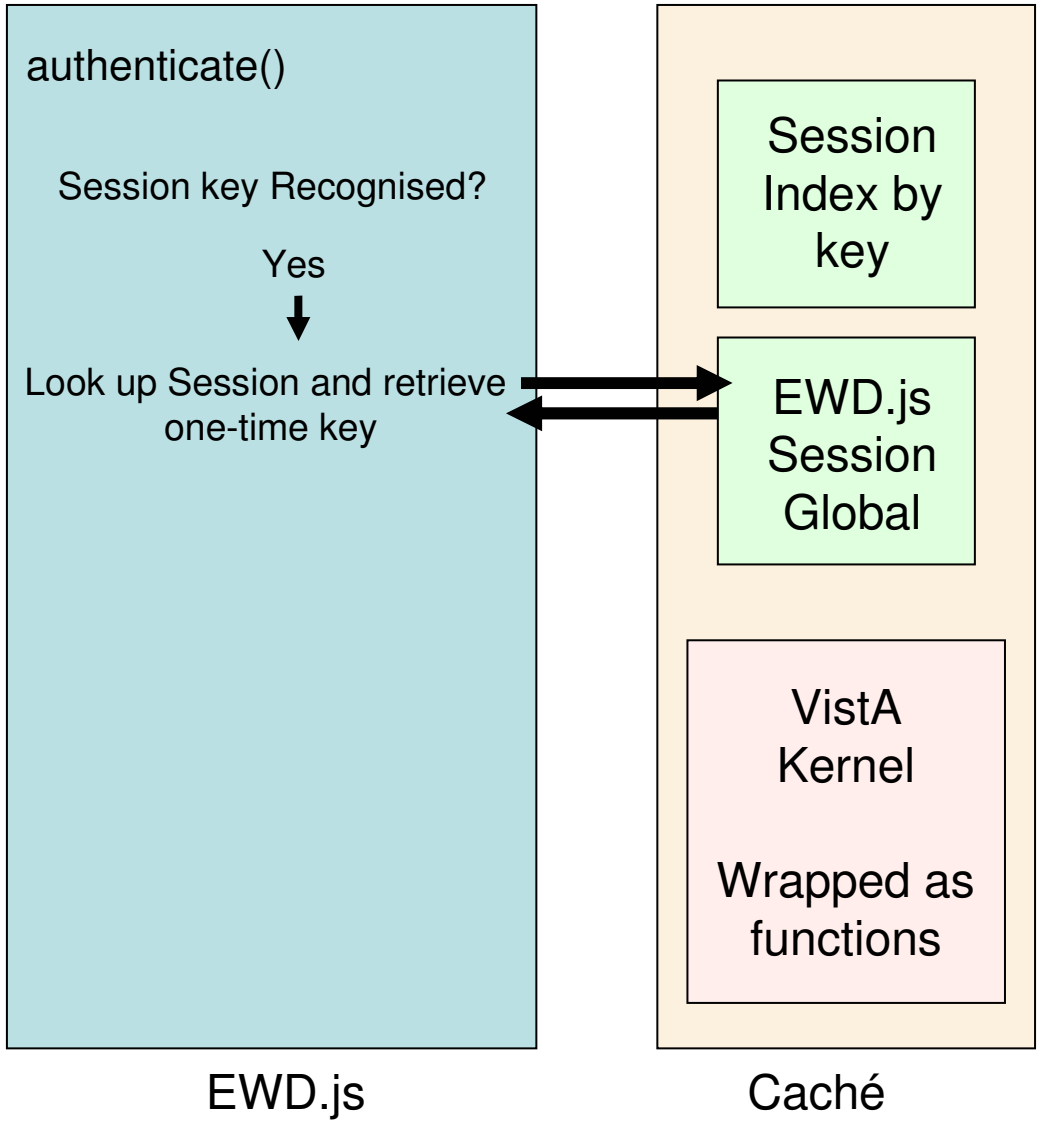
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



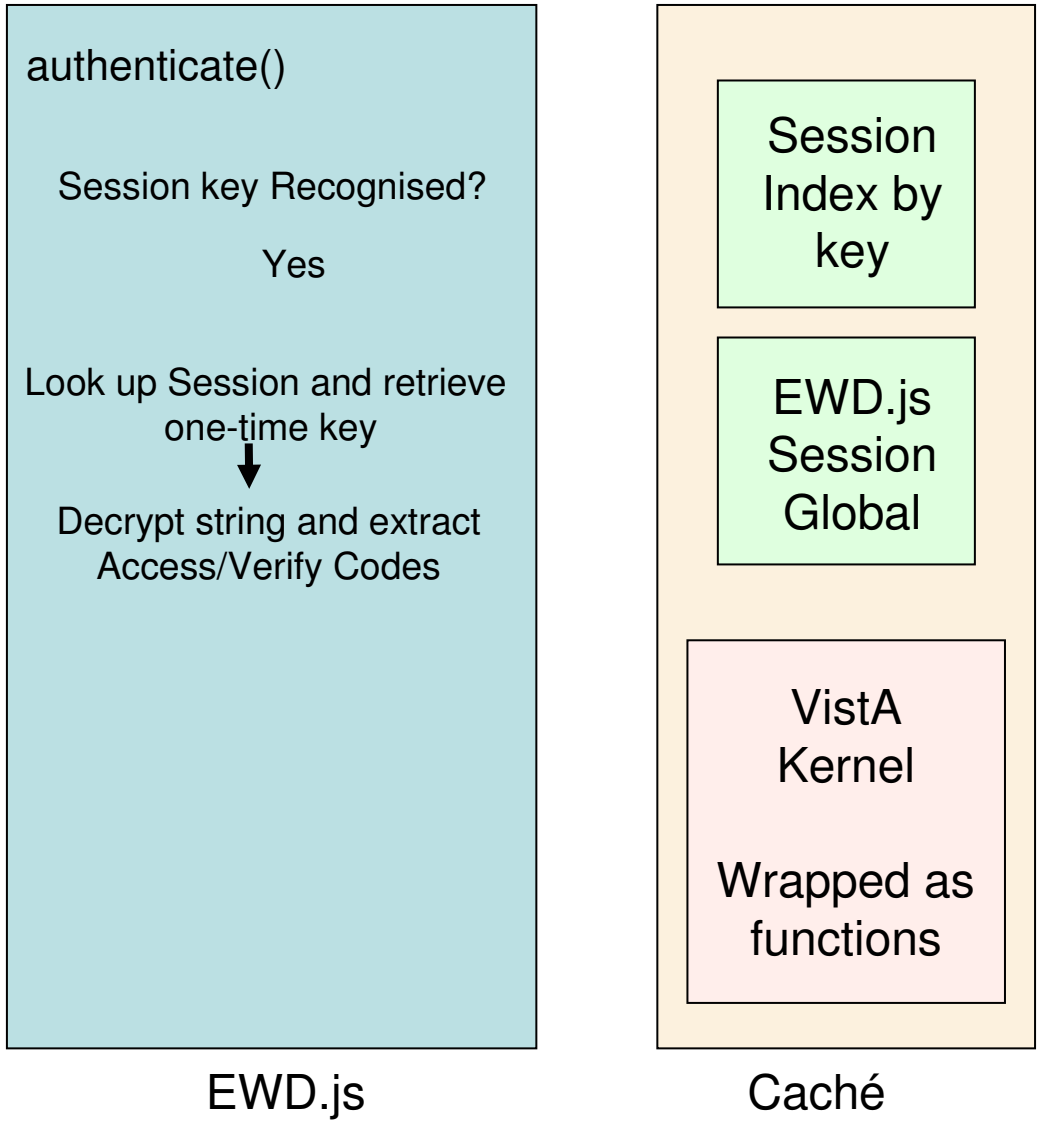
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



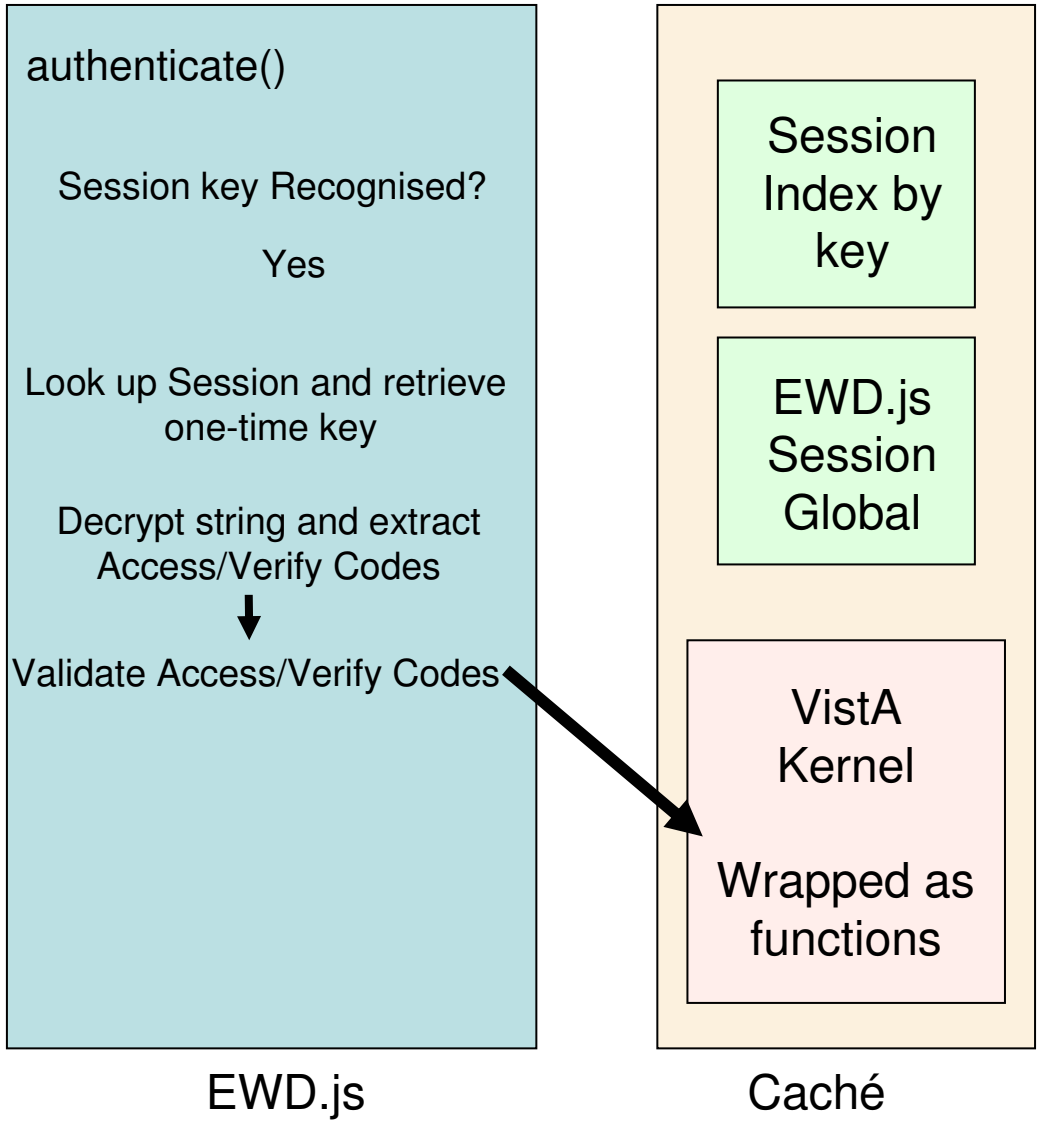
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



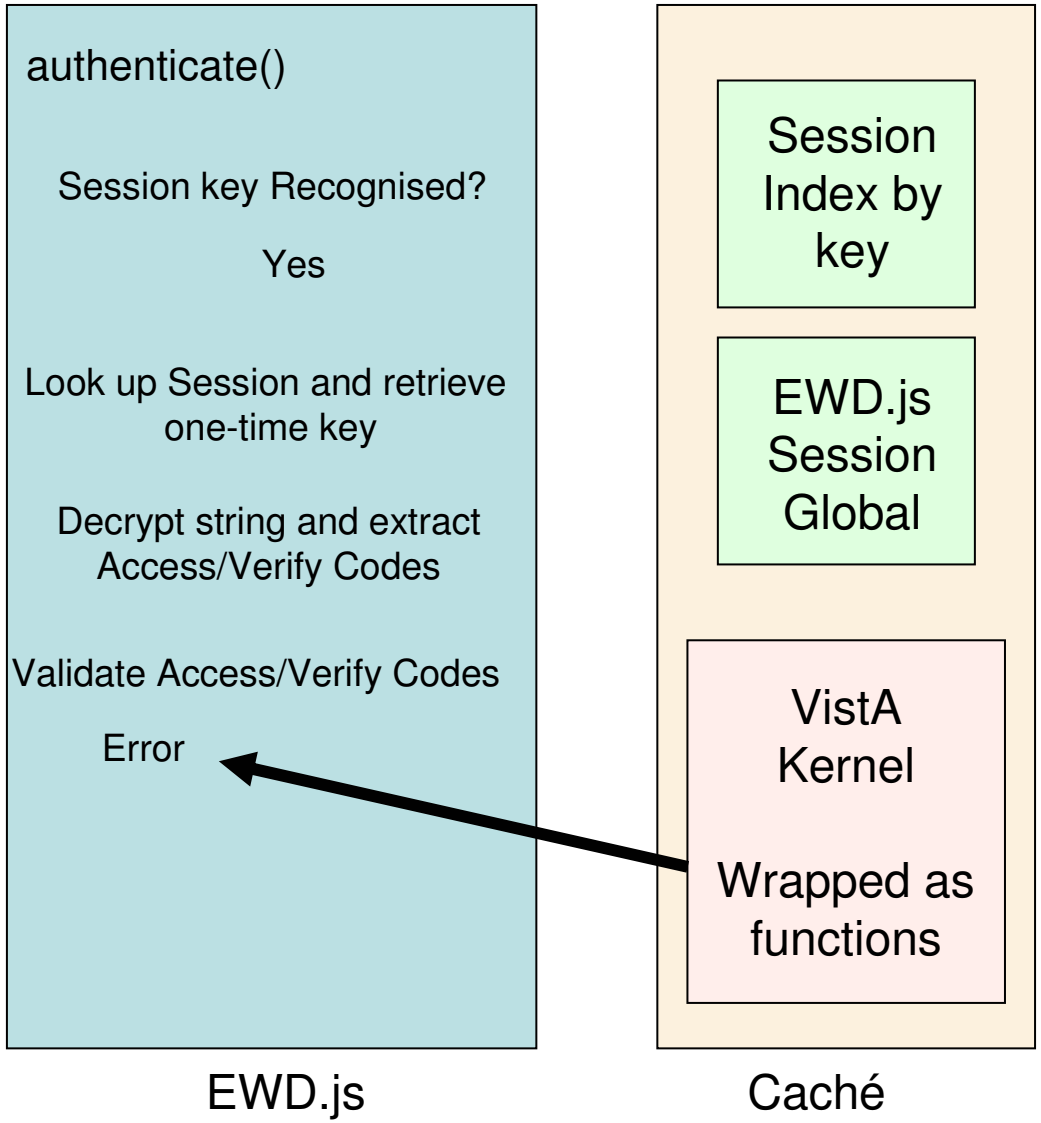
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



authenticate

Method Name

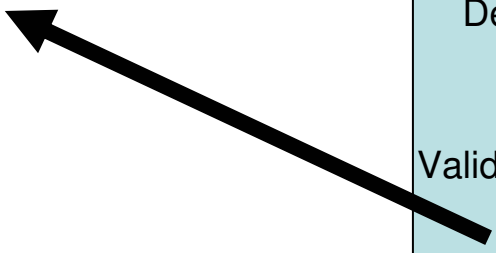
authenticate

credentials=: encrypted string

Authorization: session key

JSON Response

401 Error



authenticate()

Session key Recognised?

Yes

Look up Session and retrieve one-time key

Decrypt string and extract Access/Verify Codes

Validate Access/Verify Codes

Error

EWD.js

Session Index by key

EWD.js Session Global

VistA Kernel

Wrapped as functions

Caché

M/GATEWAY DEVELOPMENTS

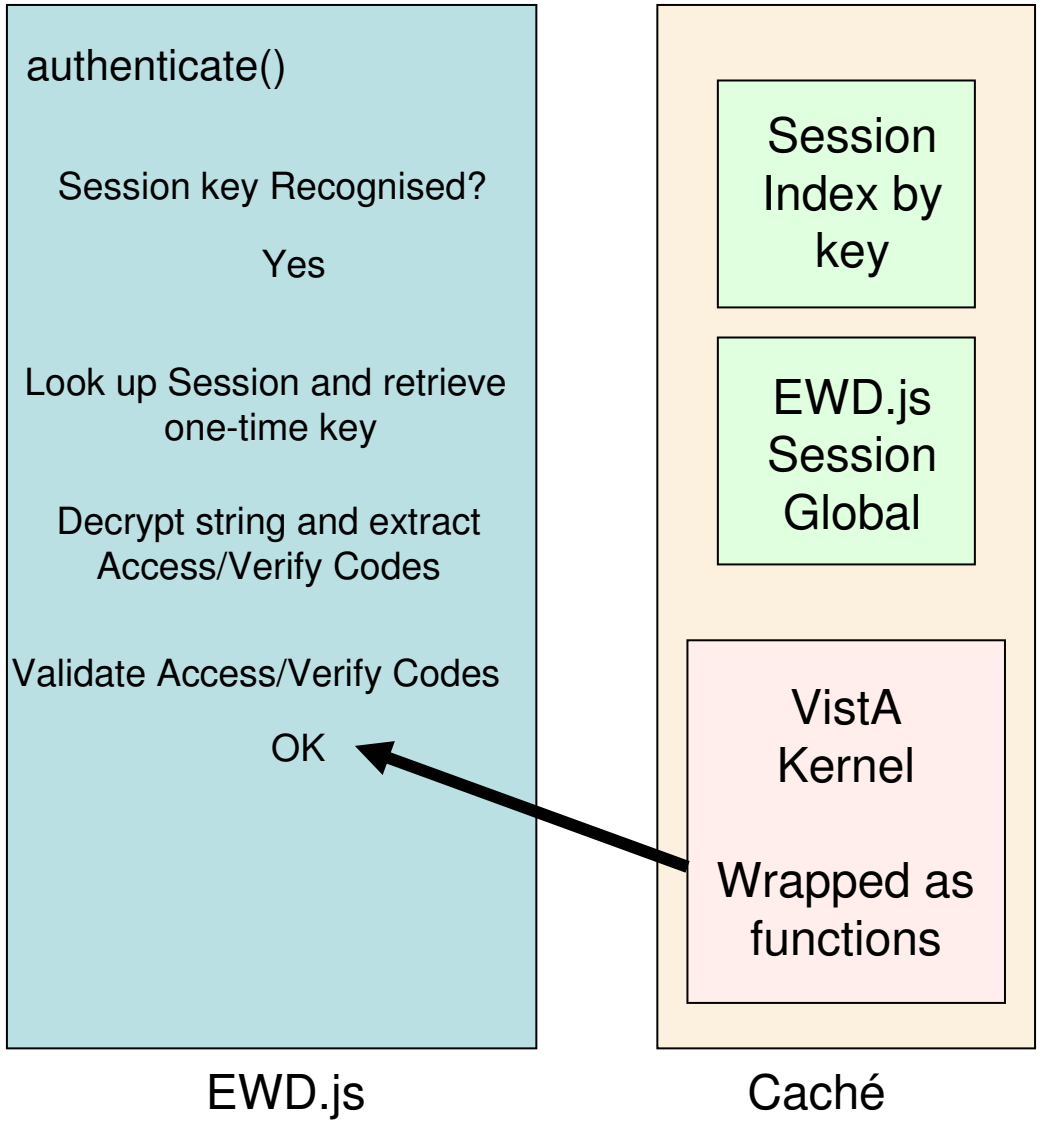
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



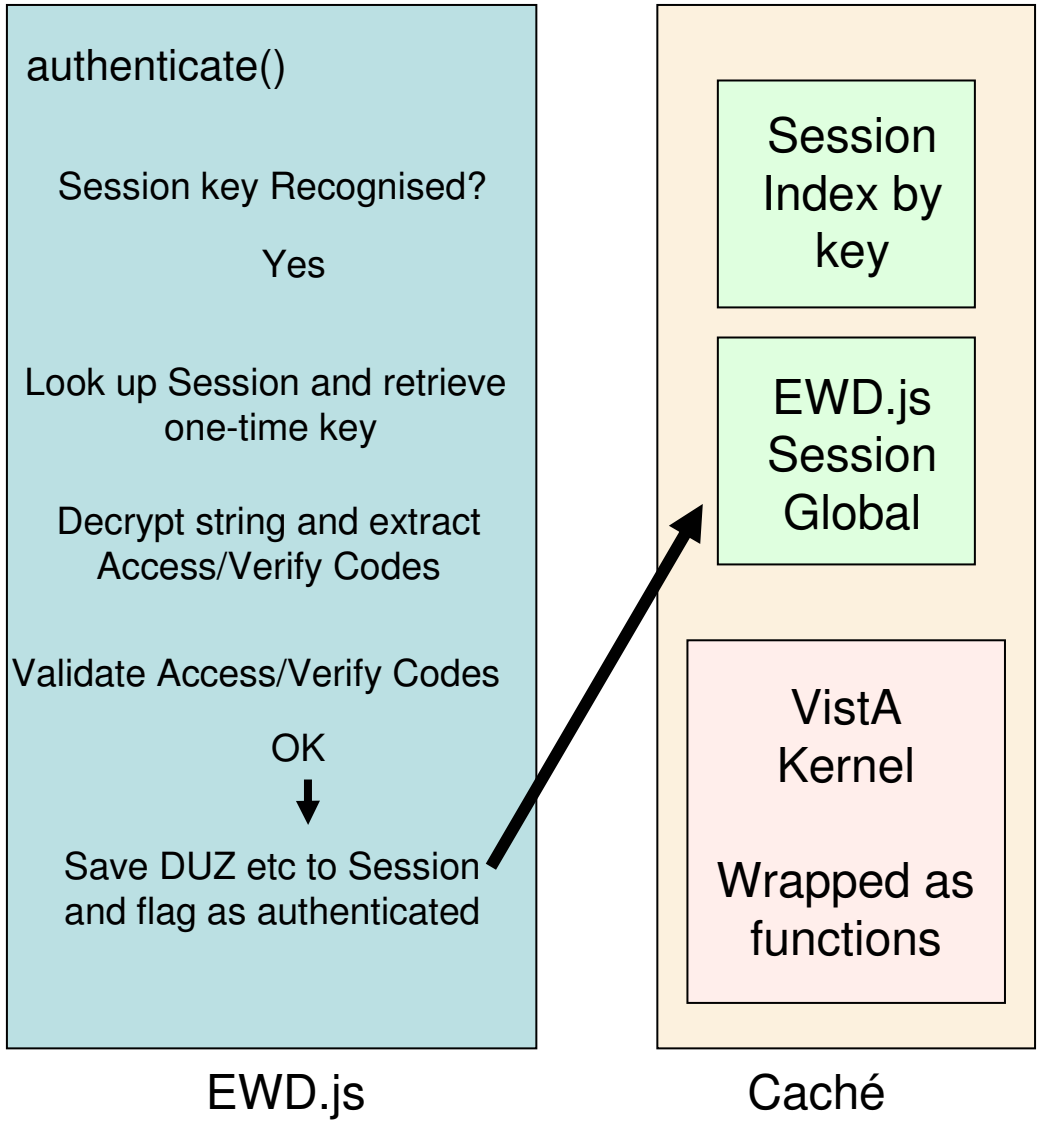
authenticate

Method Name

authenticate

credentials=: encrypted string

Authorization: session key



authenticate

Method Name

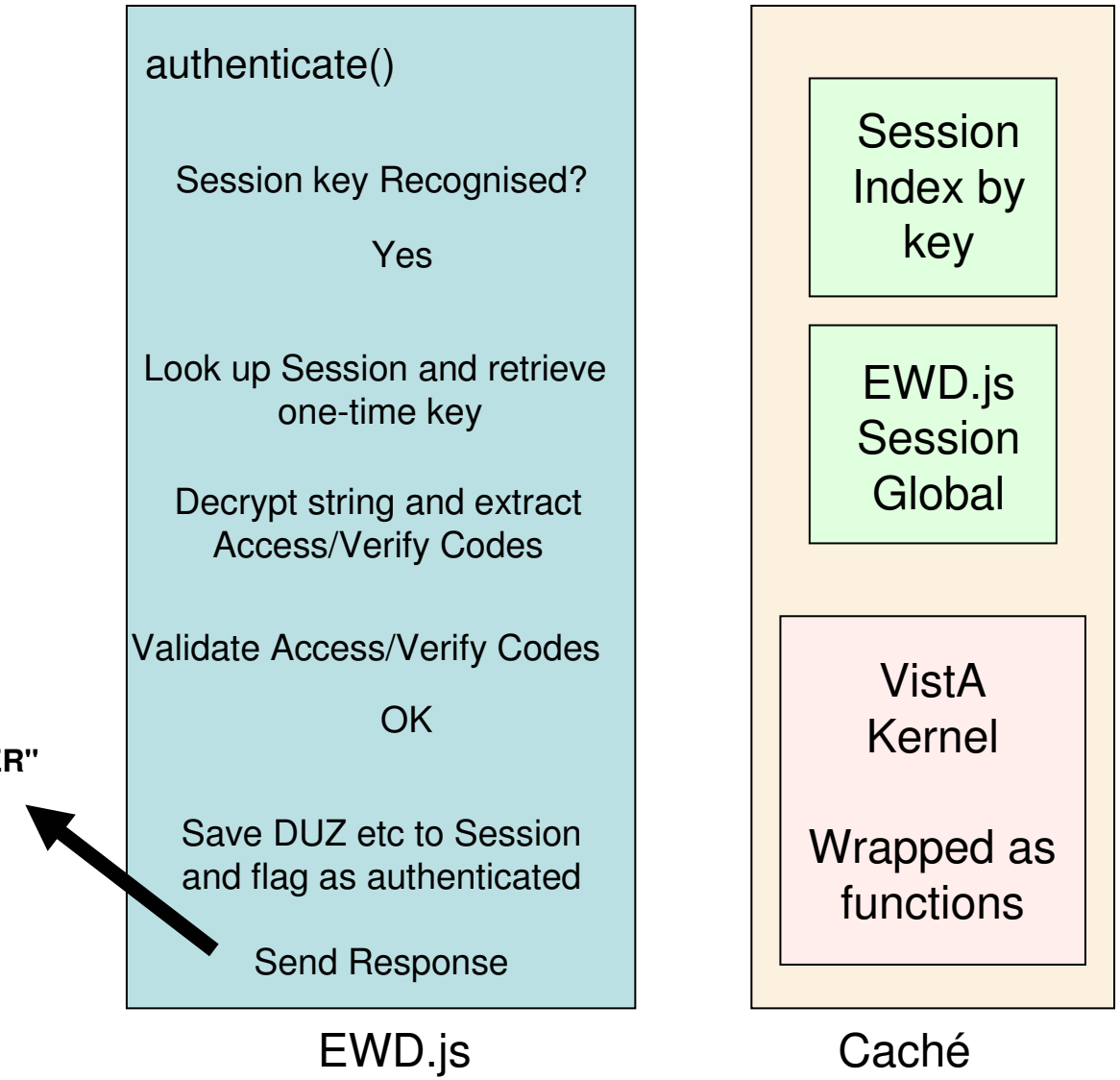
authenticate

credentials=: encrypted string

Authorization: session key

Response:

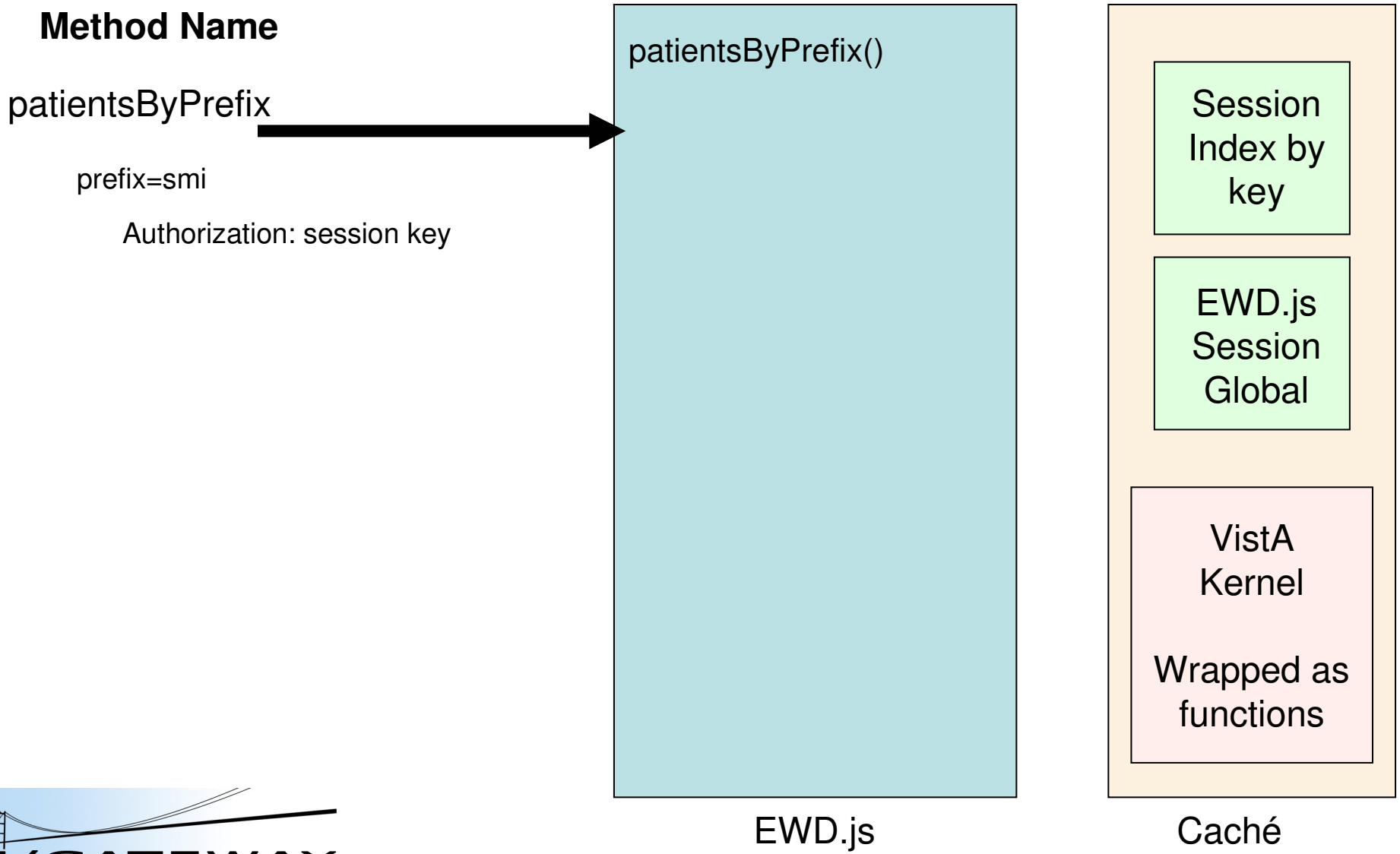
```
{
DT: "3140410"
DUZ: "991"
displayName: "DEMONSTRATION PROVIDER"
greeting: "Good morning DEMO"
username: "PROVIDER,DEMONSTRATION"
}
```



All Other Methods

- Follow a common pattern
  - Authorization Header continues to use EWD Session Token for identification
  - EWD Session cannot be used until authenticate method completes
    - Controlled by EWD Session flag variable
  - Use EWD Session to store VistA variables needed to recreate symbol table environment for VistA Kernel functions/APIs
  - Query string name/value pairs will be method-specific

All Other methods



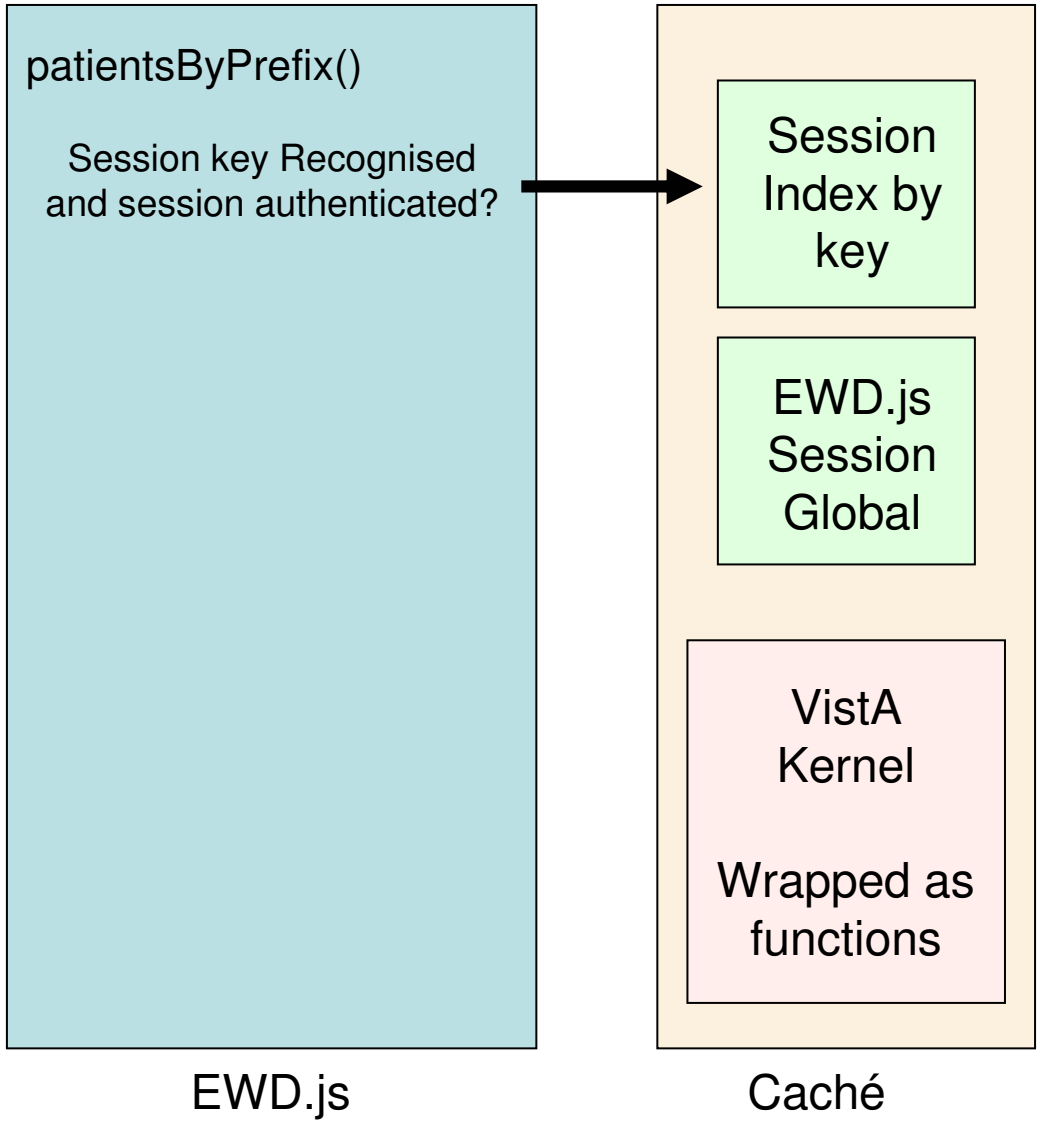
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



All Other methods

Method Name

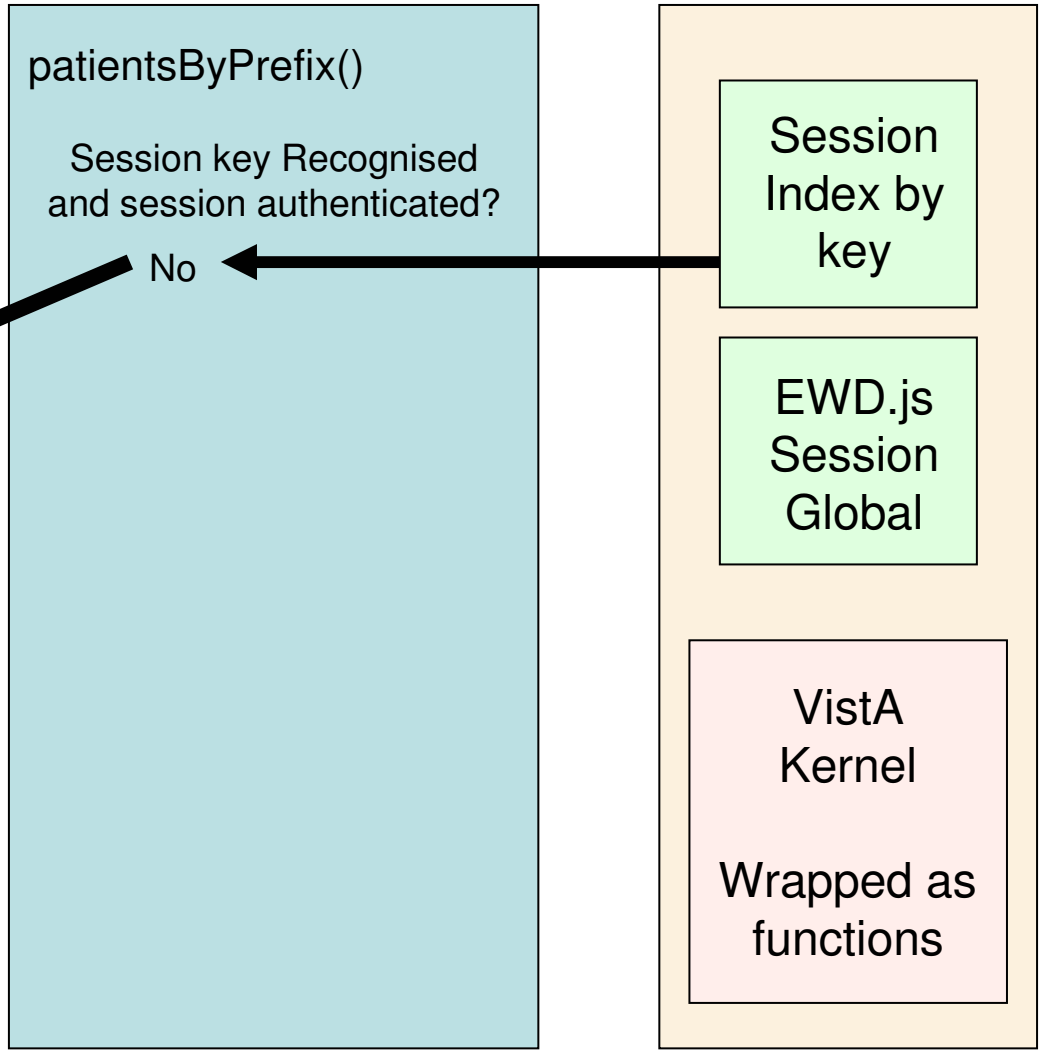
patientsByPrefix

prefix=smi

Authorization: session key

JSON Response

401 Error



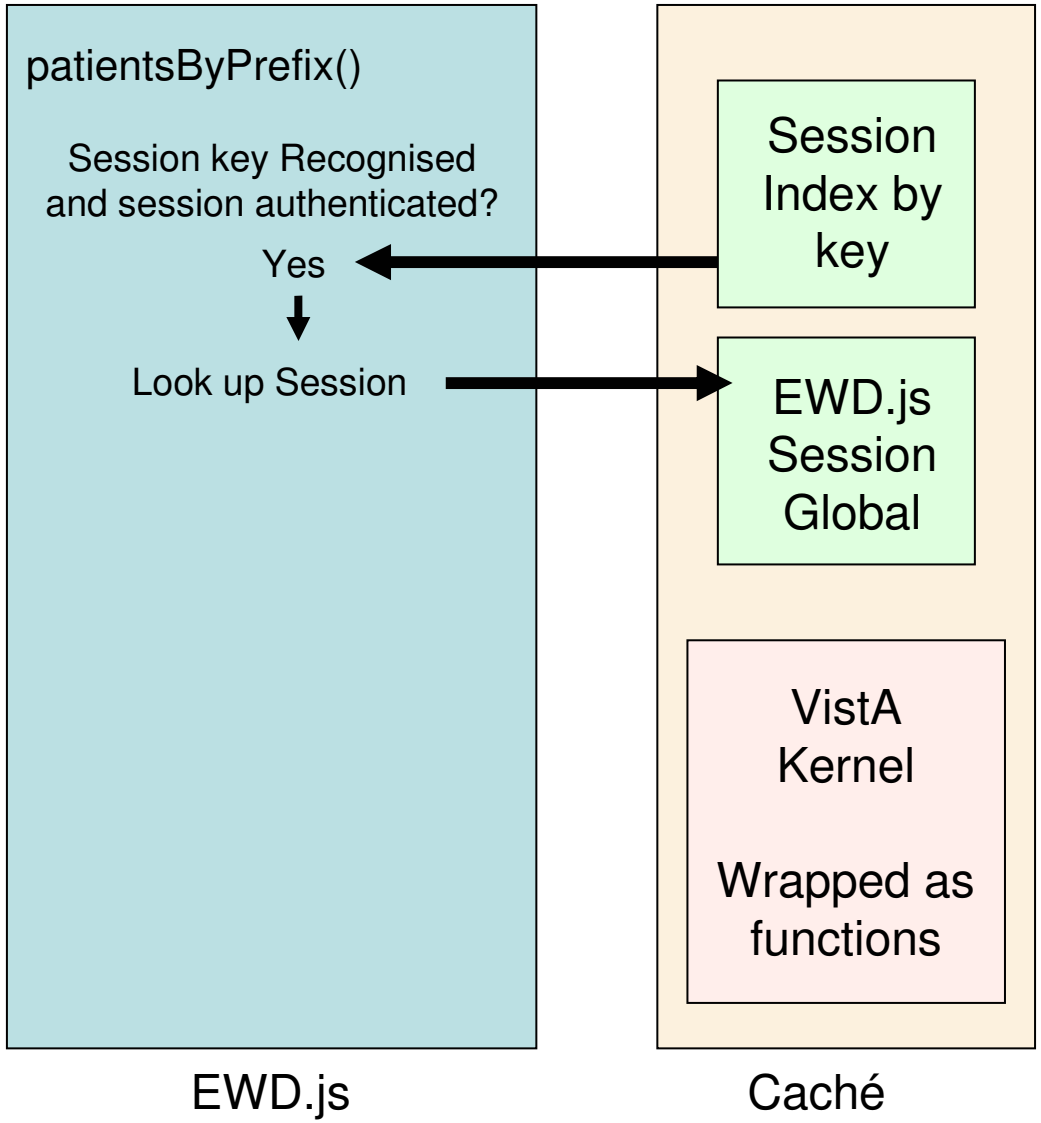
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



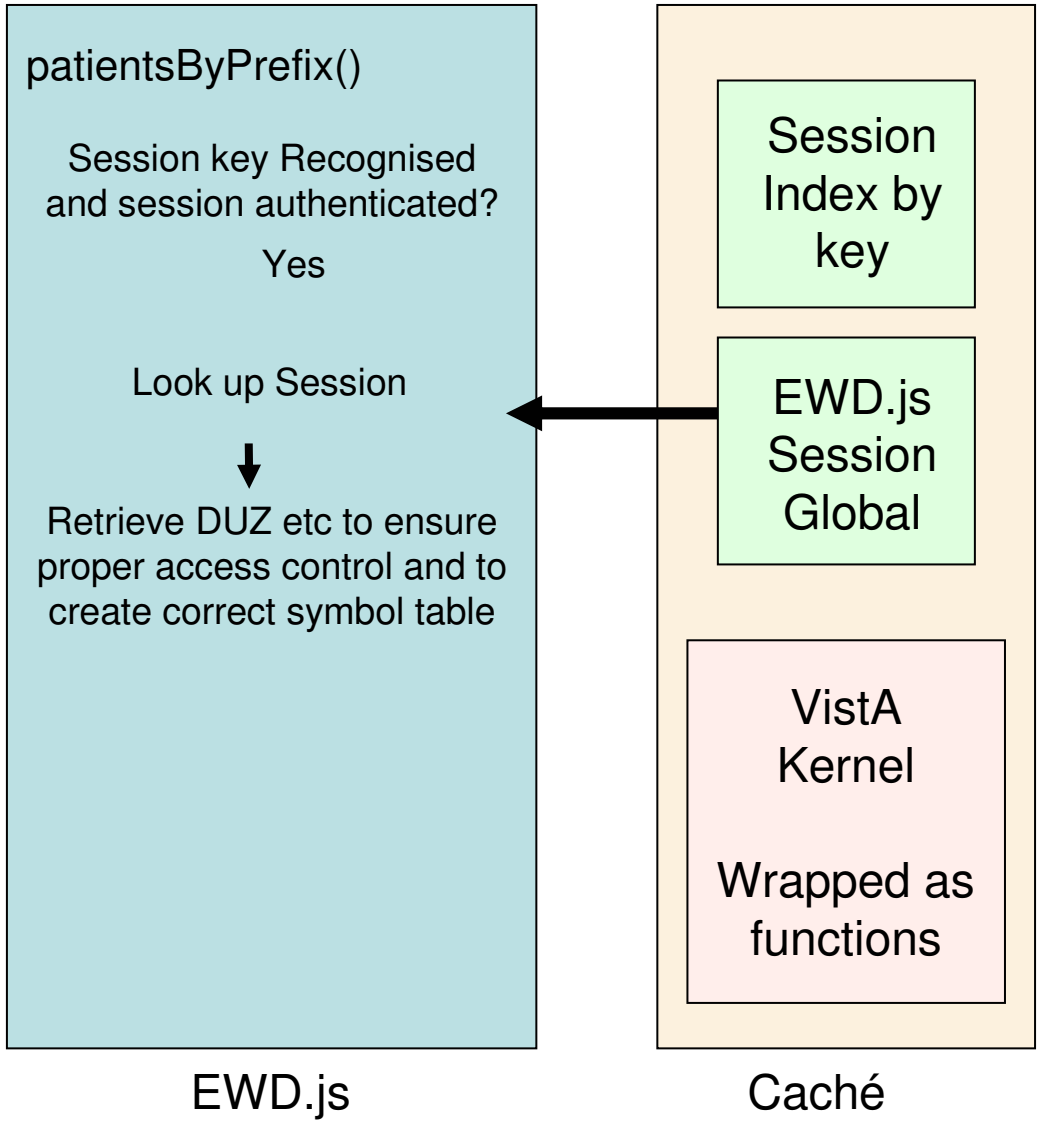
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



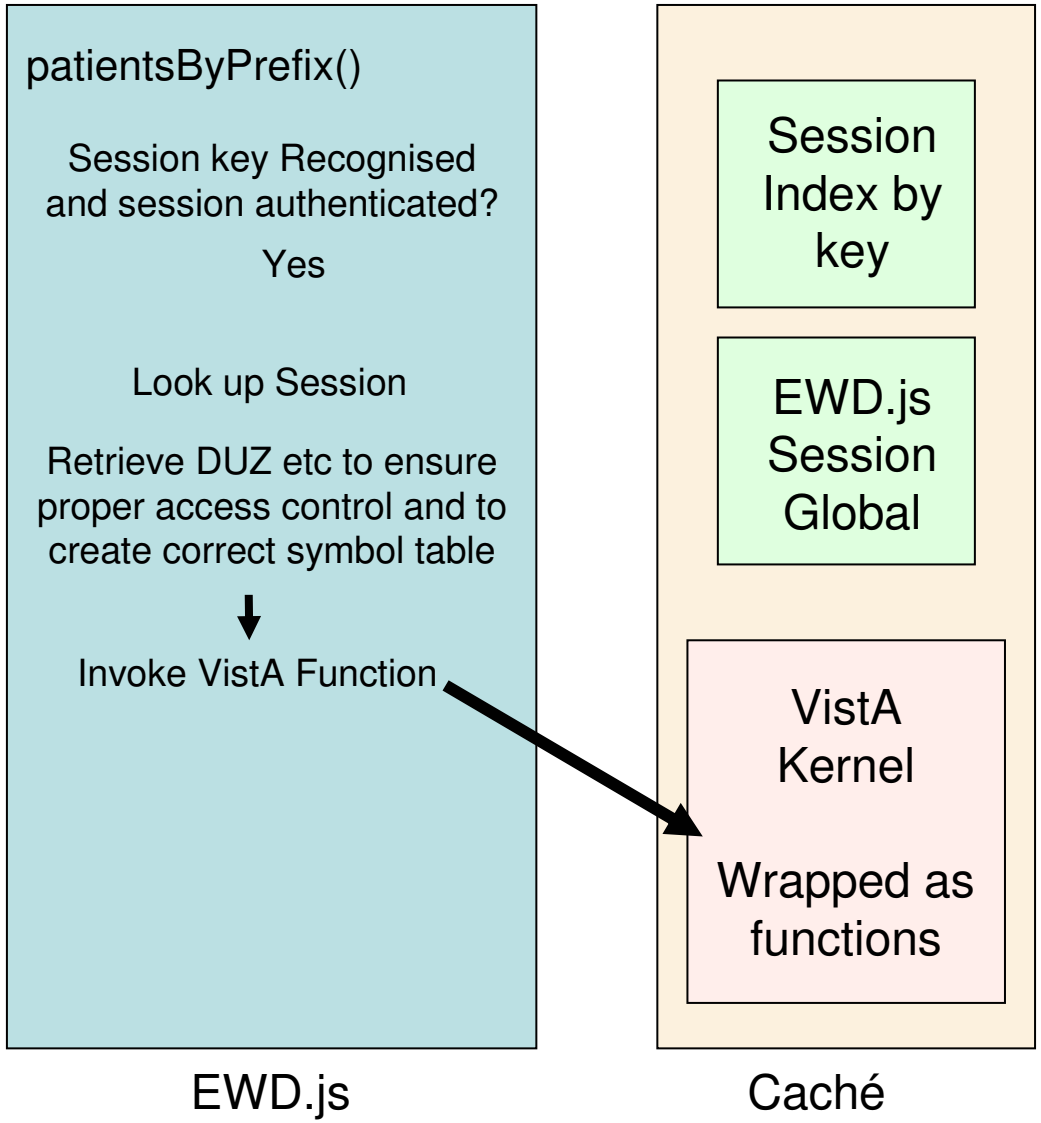
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



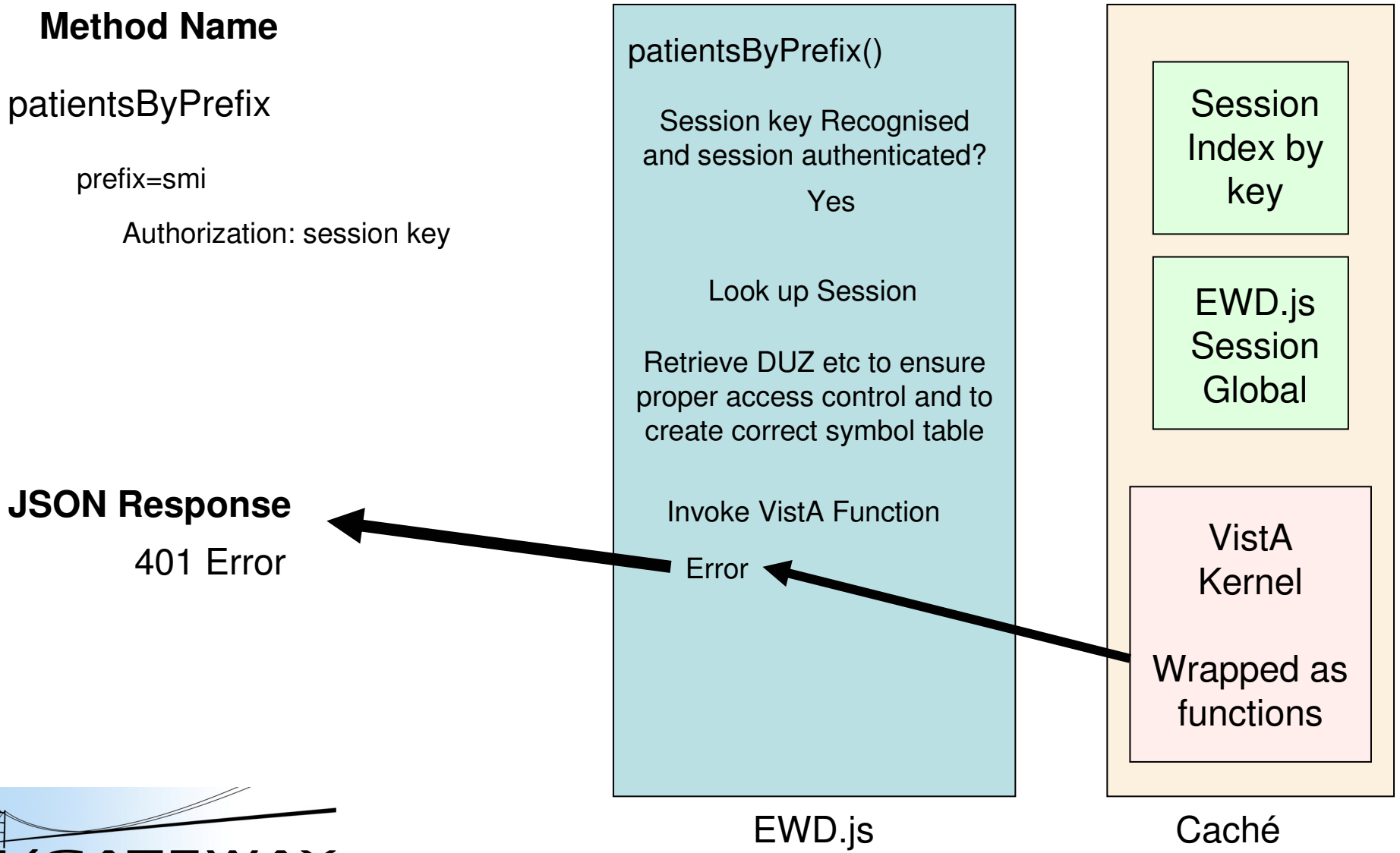
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



JSON Response

401 Error

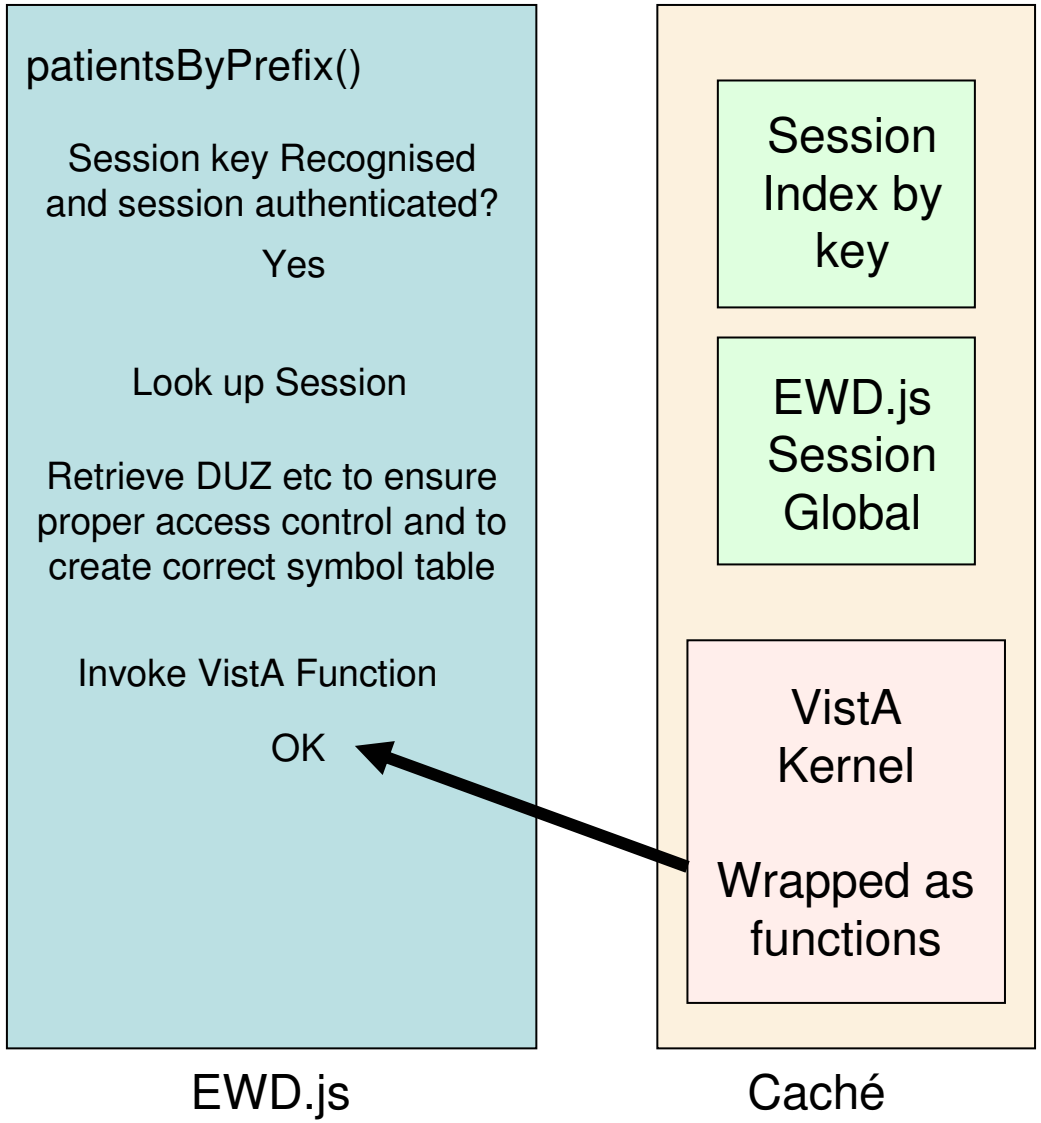
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



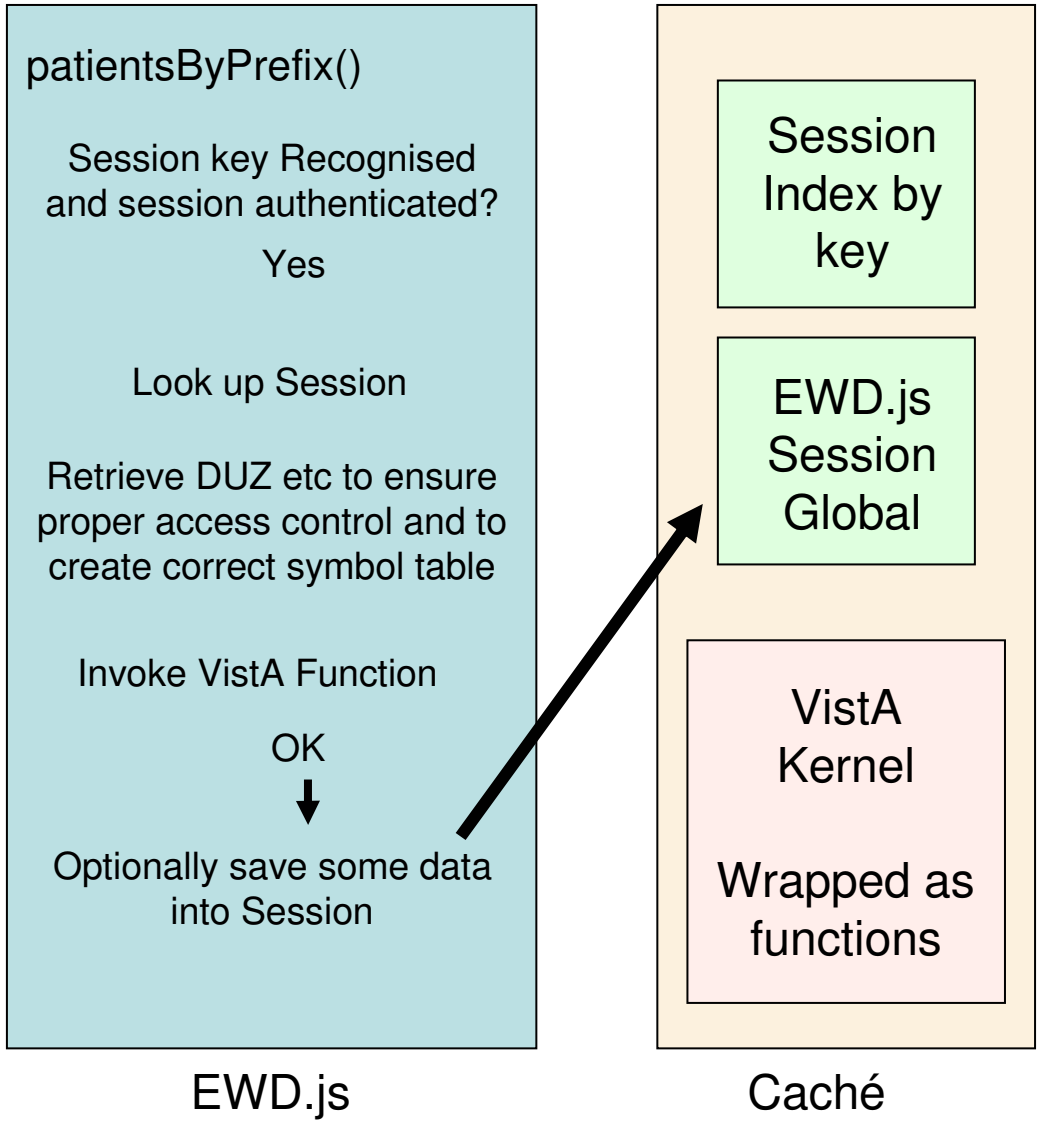
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



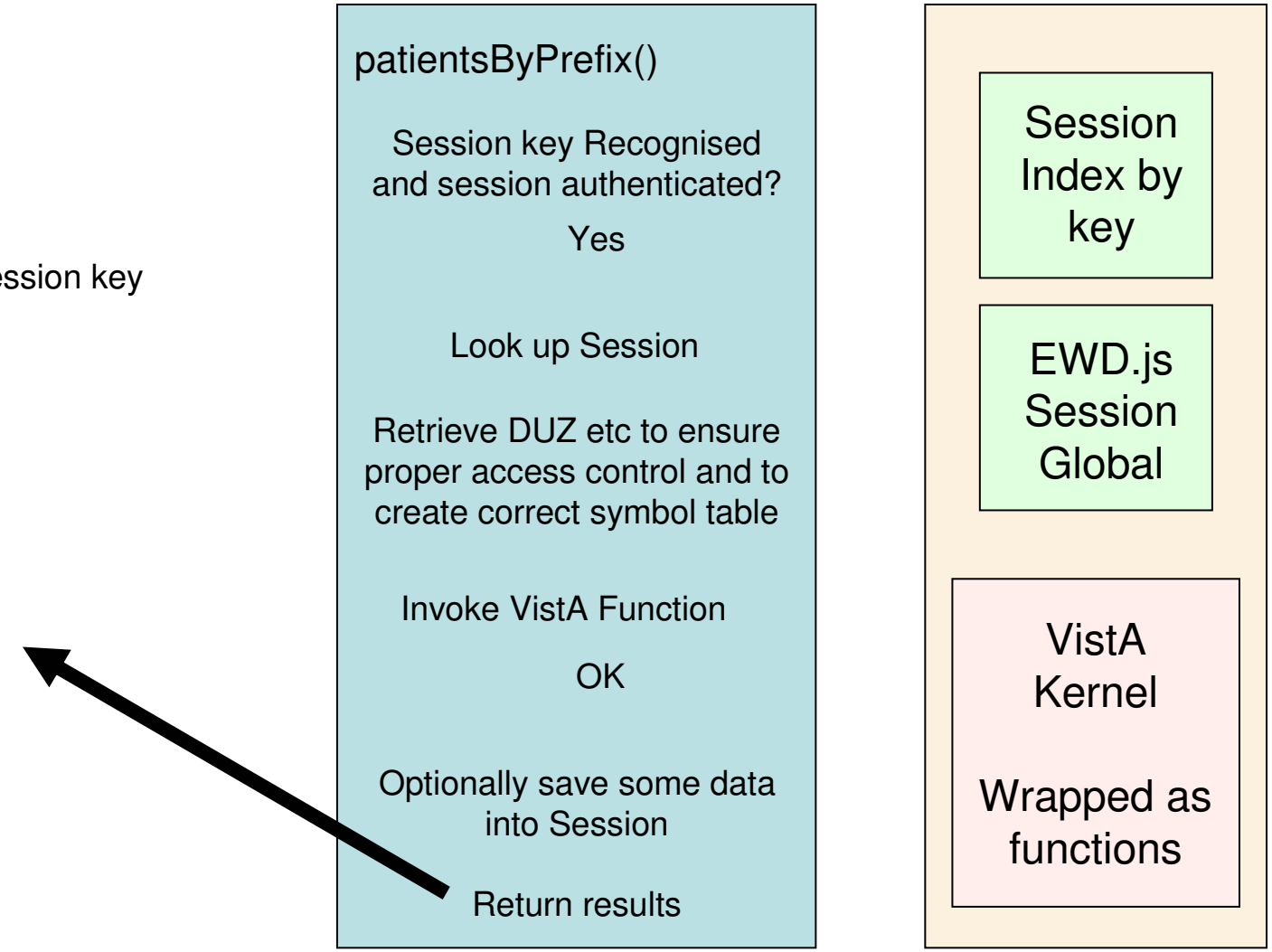
All Other methods

Method Name

patientsByPrefix

prefix=smi

Authorization: session key



Results object: JSON

EWD.js

Caché

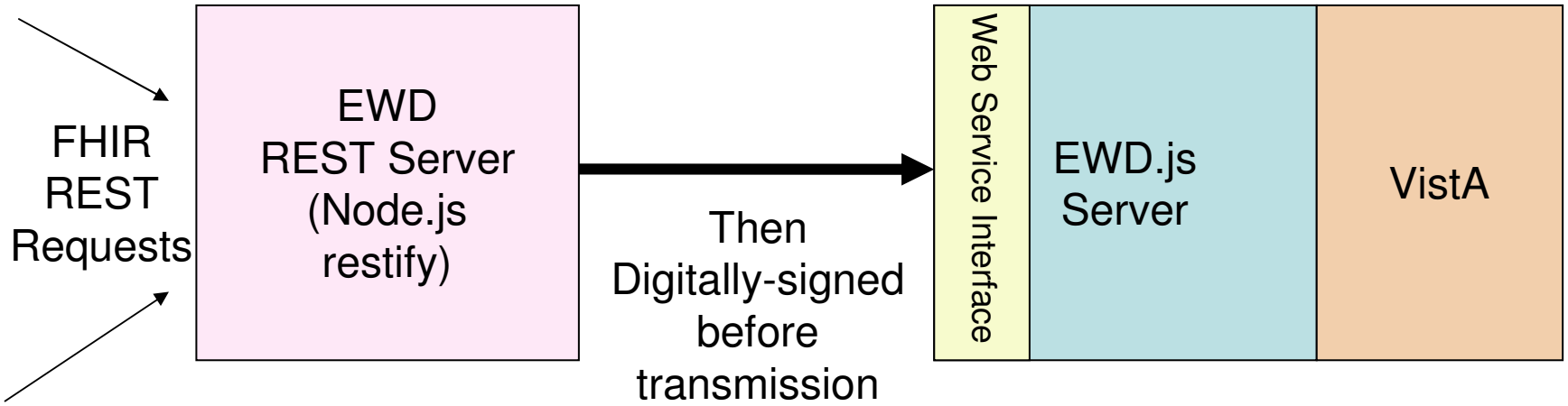
Already implemented

- See VistADemo application that is included when EWD.js is installed
  - Ready to run if you use Chris Edwards' OSEHRA installer
  - All source code provided:
    - ~/ewdjs/node_modules/VistADemo.js

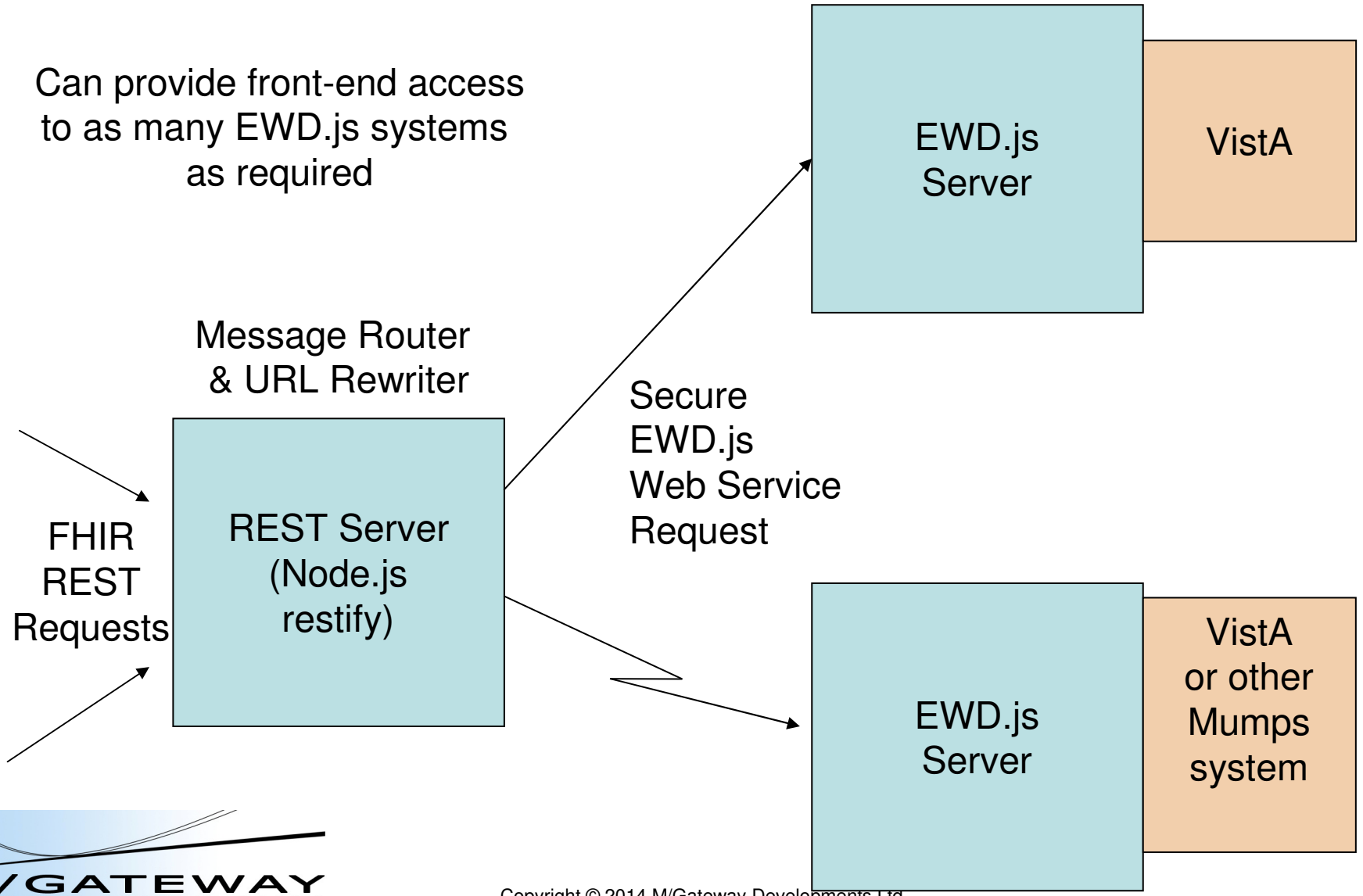
EWD REST Server

URL Re-writing:

REST requests rewritten
As EWD.js HTTP Web
Service Requests



EWD REST Server

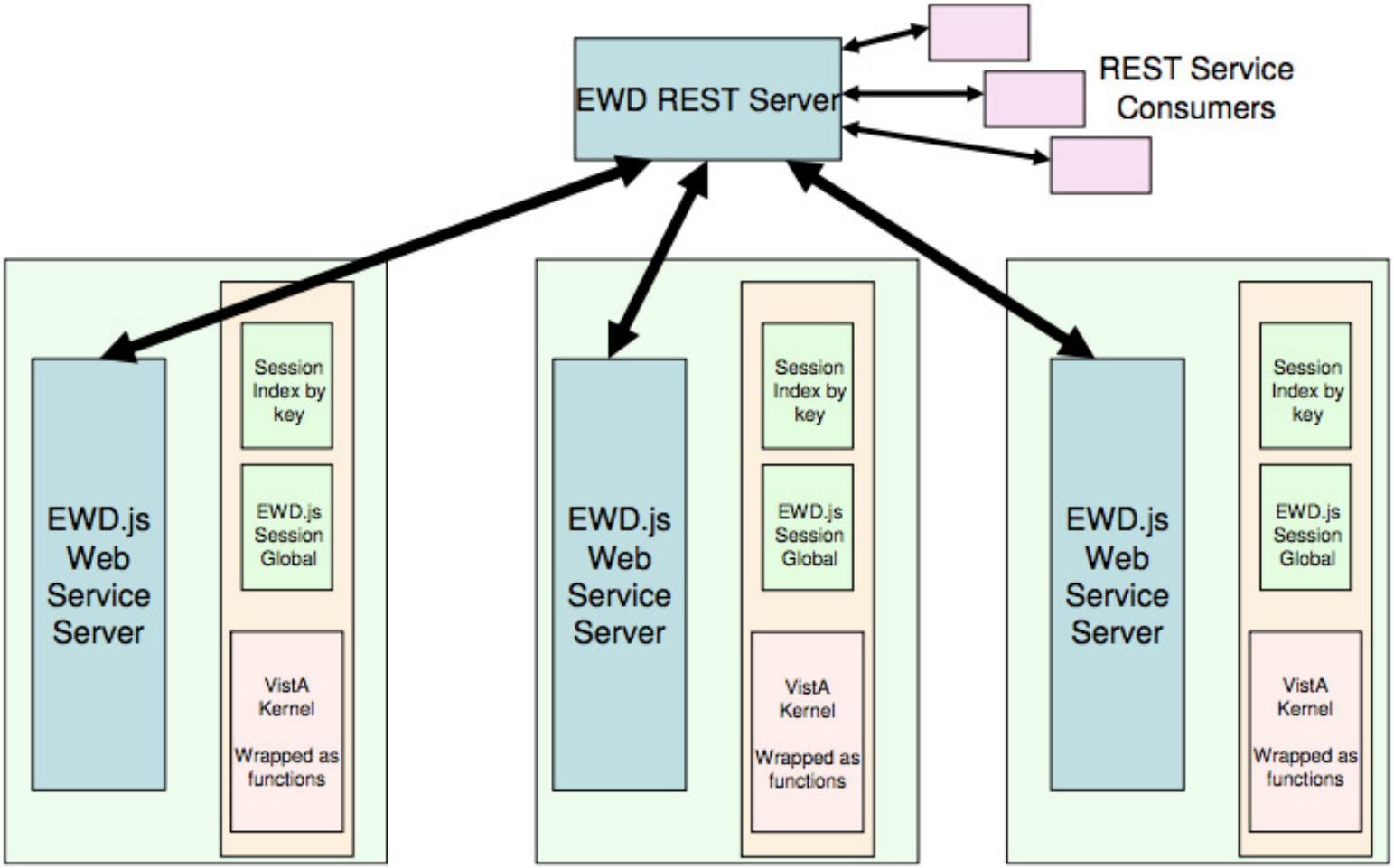


URL Re-Router

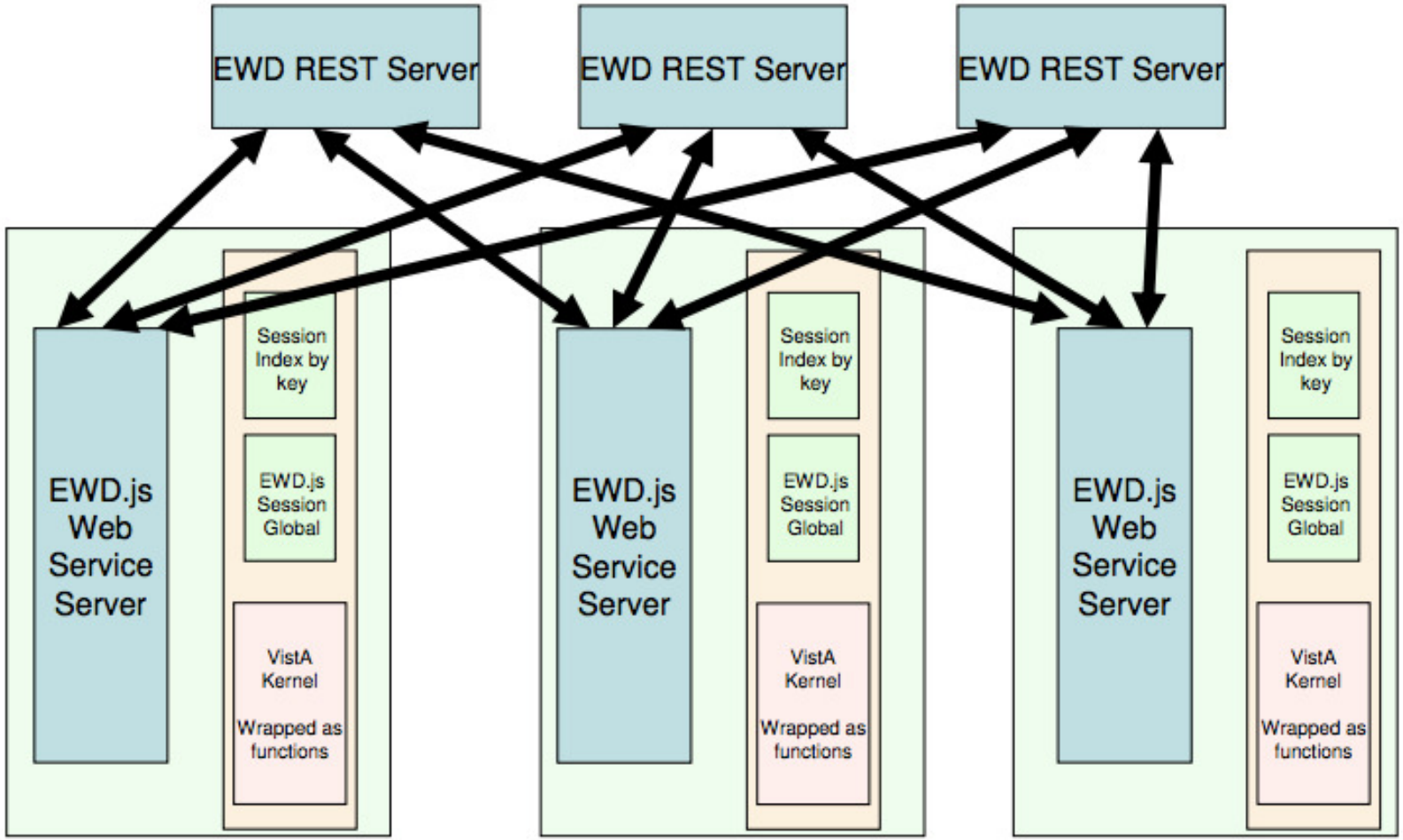
- Get Observations for patient 1234 from EC2 server
- http://192.168.1.2:8081/fhir/**ec2**/patient/@1234/observation

- Get Observations for patient 1234 from Oroville server
- http://192.168.1.2:8081/fhir/**oroville**/patient/@1234/observation

Federation using EWD.js & EWD REST Server



Redundant Configuration



The Vision in 1 slide