



Core Logic V10.3

Roel Schreibers | Jan Tytgat | Andreas Petzel



Hosting and NetScaler

- How to manage all our customer integrations?
- How to cut cost on onboarding and changes?
- How to be flexible enough to support customer demand?
- How to not overcomplicate troubleshooting by support?
- How to get a good and clear cost-model for our customers?
- What is the surplus value of a NetScaler to the customer?



Core Logic

- NetScaler does not need Core Logic per se
- Core Logic uses existing features and capabilities of the platform
- Core Logic helps standardizing the configuration



Standardization is key

- Clear the path for automation
- Cost-Model
- Support
 - Not everyone can be a NetScaler specialist (Some people have lives...)
 - Only blame it on the NetScaler if it is the NetScaler
- Change management
- DTAP (versioning)

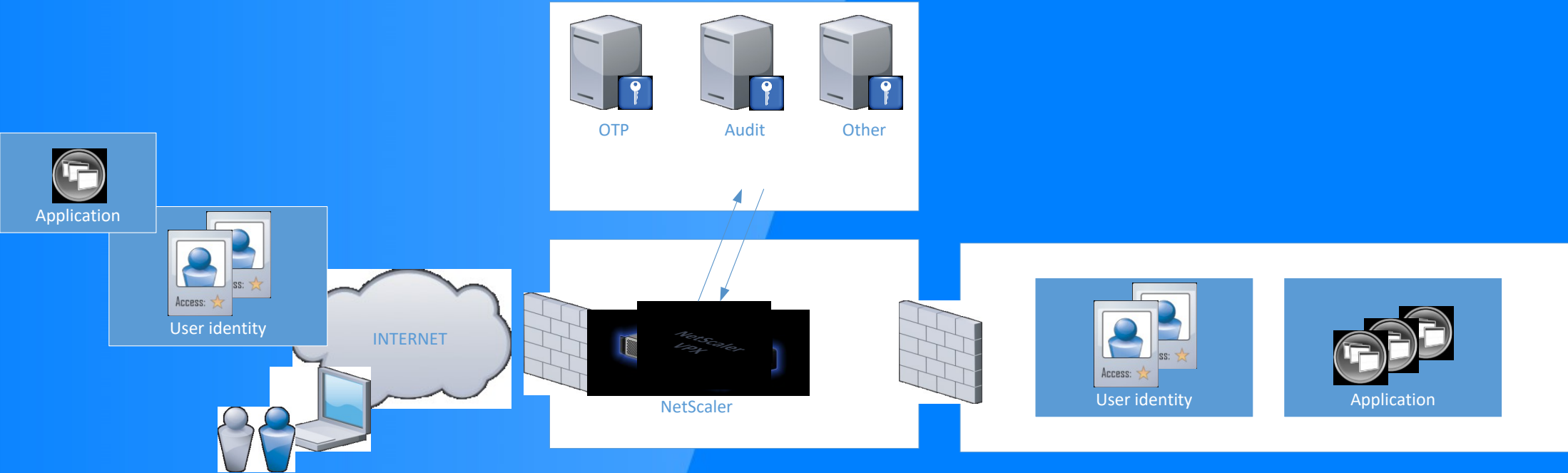


Strategic component

- The NetScaler should be regarded as a strategic component
 - Enabling business for the customer
 - Securing information
 - (Value-add optimization)
- Where does the NetScaler fit in?
 - A key component within the DMZ infrastructure
 - North-South traffic
 - Focus on optimization and security
 - A value-add component within the Server infrastructure
 - East-West traffic
 - Focus on performance



Identity and applications (data)





Communicate

- Management
- Security Officer
- Network Architect
- Developer
- Network engineer
- User / application owner
- Everyone who wants to listen

Who is more important to the enterprise than ever?



Network integration

End Users

Inbound, untrusted DMZ.
(Consolidate IP-V4 addresses)

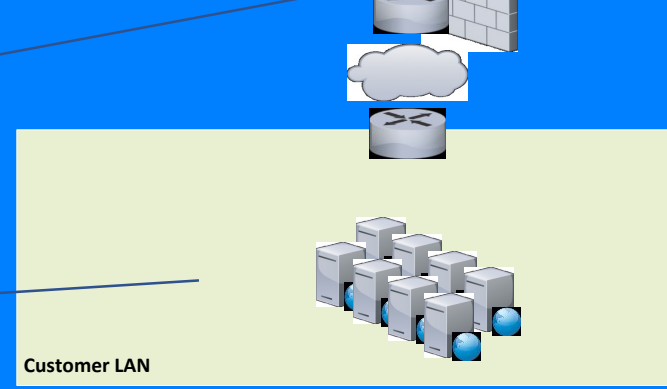
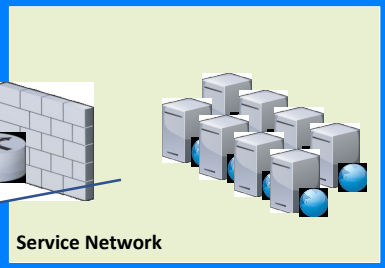
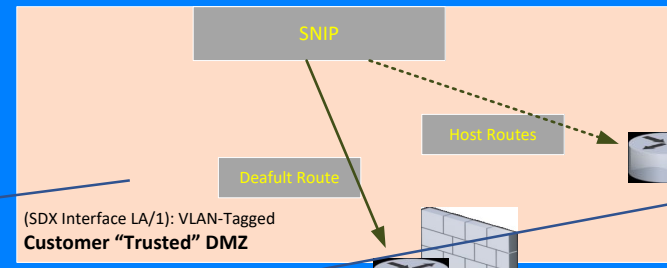
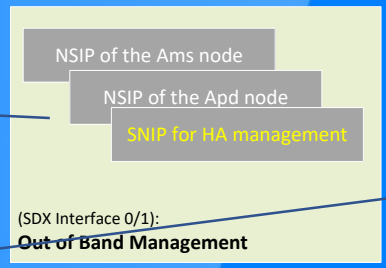
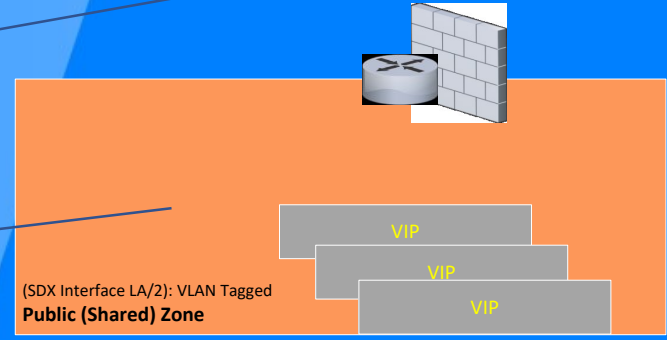
NetScaler (HA) has 3 "legs"

Out Of Band management network

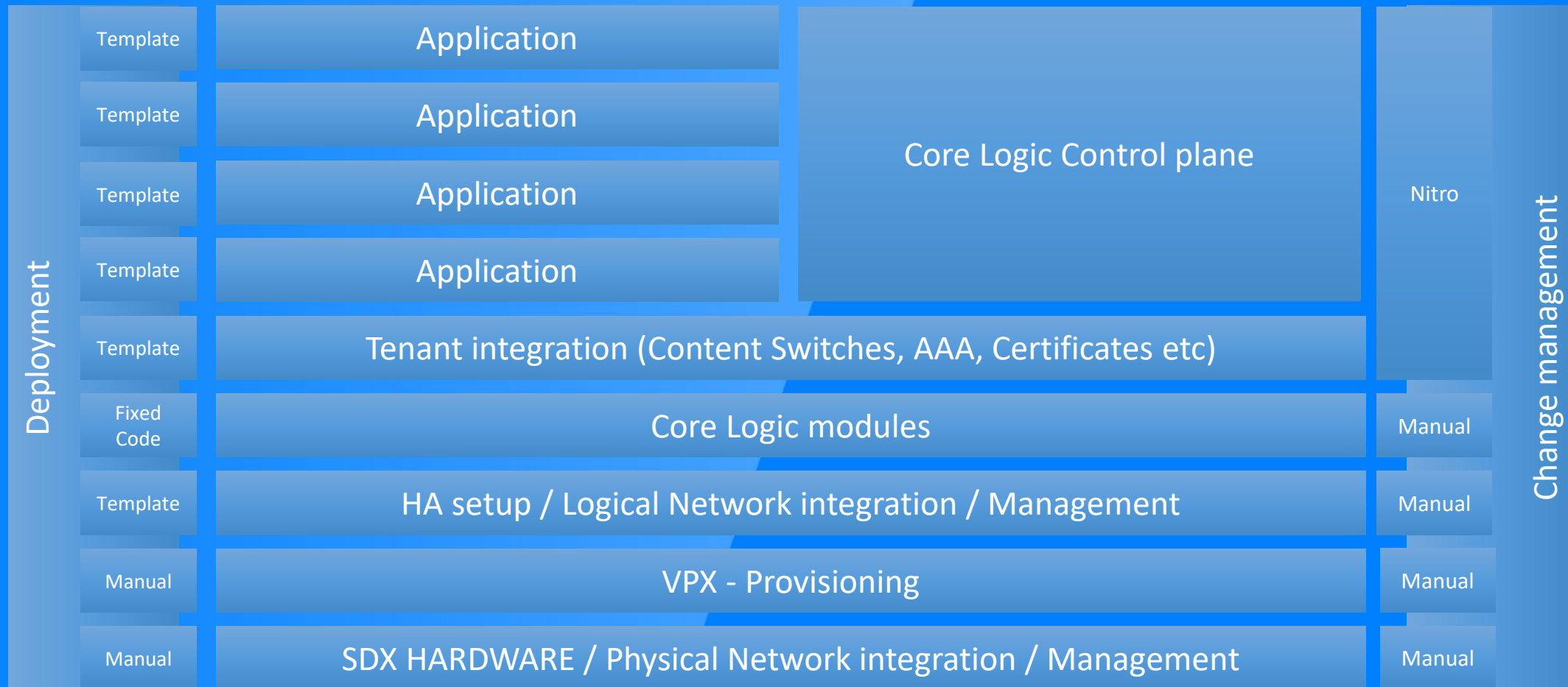
Trusted DMZ

Value added services

Customer Resources



Deployment steps



Simply put...



Applications

Customer (tenant)

Service Provider

Simply put...



Applications

CORE LOGIC

Customer (Tenant)



Demo: setup Core Logic

- Create 2 html import pages
 - Resppage_no_service
 - Resppage_blocked
- Copy | paste into the command line.
- Save ns config



Demo: setup a Tenant (SSL)

```
add cs vserver CS_T1_SSL SSL 185.63.69.210 443
bind cs vserver CS_T1_SSL -policyName CSP_FULL_LAN -priority 100
bind cs vserver CS_T1_SSL -policyName CSP_FRST_LAN -priority 101
bind cs vserver CS_T1_SSL -policyName CSP_FQDN_LAN -priority 102
bind cs vserver CS_T1_SSL -policyName CSP_WILD_LAN -priority 103
bind cs vserver CS_T1_SSL -policyName CSP_FULL_DEF -priority 110
bind cs vserver CS_T1_SSL -policyName CSP_FRST_DEF -priority 111
bind cs vserver CS_T1_SSL -policyName CSP_FQDN_DEF -priority 112
bind cs vserver CS_T1_SSL -policyName CSP_WILD_DEF -priority 113
bind cs vserver CS_T1_SSL -lbserver VS_NOSVC_SSL
```

Bind the Certificate....



Demo: setup a Tenant (HTTP)

```
add cs vserver CS_T1_HTTP HTTP 185.63.69.210 80
bind cs vserver CS_T1_HTTP -policyName CSP_FULL_LAN -priority 100
bind cs vserver CS_T1_HTTP -policyName CSP_FRST_LAN -priority 101
bind cs vserver CS_T1_HTTP -policyName CSP_FQDN_LAN -priority 102
bind cs vserver CS_T1_HTTP -policyName CSP_WILD_LAN -priority 103
bind cs vserver CS_T1_HTTP -policyName CSP_FULL_DEF -priority 110
bind cs vserver CS_T1_HTTP -policyName CSP_FRST_DEF -priority 111
bind cs vserver CS_T1_HTTP -policyName CSP_FQDN_DEF -priority 112
bind cs vserver CS_T1_HTTP -policyName CSP_WILD_DEF -priority 113
bind cs vserver CS_T1_HTTP -lbserver VS_NOSVC_HTTP
```



Demo: setup an Application

##Backend

```
add server T1_Server1 172.16.32.250
```

```
add serviceGroup SG_T1_WEB1_HTTP HTTP
```

```
bind serviceGroup SG_T1_WEB1_HTTP T1_Server1 80
```

#FrontEnd

```
add lb vserver VS_T1_WEB1_HTTP HTTP 0.0.0.0 0 -timeout 0 -cltTimeout 180
```

```
bind lb vserver VS_T1_WEB1_HTTP SG_T1_WEB1_HTTP
```



Demo: publish Application

- Bind stringmap SM_CS_CONTROL “cs_t1_ssl;www.tenant.com” “vs=VS_T1_WEB1_HTTP;”
 - *publishes the application on cs_t1_ssl through www.tenant1.com*
- Bind stringmap SM_CS_CONTROL “cs_t1_ssl;tenant.com” “vs=VS_REDIR_301;dst=//www.tenant.com;”
 - *Redirects tenant.com to www.tenant.com on cs_t1_ssl*
- Bind stringmap SM_CS_CONTROL “cs_t1_ssl;www.tenant.com/” “vs=VS_REDIR_302;dst=/logon;”
 - *Redirects www.tenant.com + empty path to www.tenant.com/logon on cs_t1_ssl*
- Bind stringmap SM_CS_CONTROL “cs_t1_ssl;www.tenant.com/wp-admin” “vs=VS_BLOCKED;”
 - *Blocks access to www.tenant.com/wp-admin on cs_t1_ssl*



Demo: Define LAN

- Bind stringmap SM_IP_CONTROL “cs_t1;10.31.0.0/16” “Sam Office Horst”
 - *Defines 10.31.0.0/16 as LAN on cs_t1_ssl and cs_t1_http*
- Bind stringmap SM_IP_CONTROL “cs_t1;145.16.123.53/32” “Developers”
 - *Defines 145.16.123.53/32 (single host) as LAN on cs_t1_ssl and cs_t1_http*
- Bind stringmap SM_CS_CONTROL “cs_t1_ssl;lan;www.tenant.com/wp-admin” “vs=VS_T1_WEB1_HTTP;”
 - *Allows access to www.tenant.com/wp-admin on cs_t1_ssl if client resides in one of the LAN segments.*



Core Logic & Web Applications

- SM_IP_CONTROL

- Regulates the IP Subnets that need be considered LAN for the Tenant
- Valid for both `_ssl` and `_http` Content Switches.

- SM_CS_CONTROL

- Regulates the flow of requests to a loadbalancing Vserver based on:
 - FQDN + FullPath
 - FQDN + FirstPath
 - FQDN
 - Wildcard domain (no path!!)
- Content Switch specific
- Distincts between **LAN** and **DEFault**

Stringmap key values should allways be in LOWERCASE !!!

Stringmaps are indexed; they can get very large without performance impact!!!



Default Loadbalancing VServers

#	Name	What it does..
1	VS_REDIR_302_SWITCH	Redirects the request from HTTP to HTTPS, or vice versa
2	VS_REDIR_302	*Redirects the request using “302 Found” (uses dst=...;)
3	VS_REDIR_301	*Redirects the request using “301 Moved Permanently ” (uses dst=...;)
4	VS_BLOCKED	Responds with the “resppage_blocked” html code
5	VS_DROP	Drops the request
6	VS_RESET	Resets the request
7	VS_NOSVC_HTTP	**default on cs_..._http... (Alamo: redirect to _ssl)
8	VS_NOSVC_SSL	**default on cs_..._ssl... (Alamo: resppage_blocked)

*VS_REDIR_301 and VS_REDIR_302 use the “dst=...;” field to calculate the redirect Location. Relative and Absolute redirects are both allowed here.

**VS_NOSVC_HTTP and VS_NOSVC_SSL will first check if there is an entry in the SM_CS_CONTROL stringmap (perhaps the backend is down). If an entry is found these Vservers will respond with the “resppage_no_service” html code.



Core Logic

IT is complicated enough

Understanding how it works is a different story...



How the west was won

- Stringmaps are indexed (fast)
- Contentswitch policies and actions can be calculated
- Responder policies and actions can be calculated
- We can concatenate anything and compare it with a Stringmap entry:
 - CSname: `HTTP.REQ.CS_VSERVER.NAME`
 - FQDN: `HTTP.REQ.HOSTNAME.SERVER`
 - WILD: `HTTP.REQ.HOSTNAME.DOMAIN`
 - FullPath: `HTTP.REQ.URL.PATH`
 - 1stPath: `HTTP.REQ.URL.PATH.GET(1)`



Check SM_CS_CONTROL keys

*Concept used in Content Switch and Responder policies

- **CHECK if LAN ENTRY**

- Check fullpath: `PE_IS_LAN && (Ccname + ";lan;" + FQDN + "/" + FullPath).IS_STRINGMAP_KEY("SM_CS_CONTROL")`
- Check 1stpath: `PE_IS_LAN && (Ccname + ";lan;" + FQDN + "/" + 1stPath).IS_STRINGMAP_KEY("SM_CS_CONTROL")`
- Check FQDN: `PE_IS_LAN && (Ccname + ";lan;" + FQDN).IS_STRINGMAP_KEY("SM_CS_CONTROL")`
- Check WILD: `PE_IS_LAN && (Ccname + ";lan;" + WILD).IS_STRINGMAP_KEY("SM_CS_CONTROL")`

- **CHECK if DEFault ENTRY**

- Check fullpath: `(Ccname + ";" + FQDN + "/" + FullPath).IS_STRINGMAP_KEY("SM_CS_CONTROL")`
- Check 1stpath: `(Ccname + ";" + FQDN + "/" + 1stPath).IS_STRINGMAP_KEY("SM_CS_CONTROL")`
- Check FQDN: `(Ccname + ";" + FQDN).IS_STRINGMAP_KEY("SM_CS_CONTROL")`
- Check WILD: `(Ccname + ";" + WILD).IS_STRINGMAP_KEY("SM_CS_CONTROL")`

**PE_IS_LAN checks if the client subnet is registered in CS_IP_CONTROL:

```
((CSName.BEFORE_STR("_ssl" or "_http") + ";" + CLIENT.IP.SRC.SUBNET(X).TYPECAST_TEXT_T + "/" + X).IS_STRINGMAP_KEY("SM_IP_CONTROL"))
```

X runs from 14 to 32....



Get SM_CS_CONTROL “vs=” value

*Concept used in Content Switch actions

- **Get if LAN ENTRY**

- Getvs fullpath: `(Ccname + “;lan;” + FQDN + “/” + FullPath).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“vs=”).BEFORE_STR(“;”)`
- Getvs 1stpath: `(Ccname + “;lan;” + FQDN + “/” + 1stPath). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`
- Getvs FQDN: `(Ccname + “;lan;” + FQDN). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`
- Getvs WILD: `(Ccname + “;lan;” + WILD). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`

- **Get if DEFault ENTRY**

- Getvs fullpath: `(Ccname + “;” + FQDN + “/” + FullPath). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`
- Getvs 1stpath: `(Ccname + “;” + FQDN + “/” + 1stPath). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`
- Getvs FQDN: `(Ccname + “;” + FQDN). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`
- Getvs WILD: `(Ccname + “;” + WILD). MAP_STRING(“SM_CS_CONTROL”). AFTER_STR(“vs=”).BEFORE_STR(“;”)`



Get SM_CS_CONTROL “dst=” value

*Concept used in Responder actions

- **Get if LAN ENTRY**

- Getdst fullpath: `(Ccname + “;lan;” + FQDN + “/” + FullPath).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`
- Getdst 1stpath: `(Ccname + “;lan;” + FQDN + “/” + 1stPath).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`
- Getdst FQDN: `(Ccname + “;lan;” + FQDN).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`
- Getdst WILD: `(Ccname + “;lan;” + WILD).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`

- **Get if DEFault ENTRY**

- Getdst fullpath: `(Ccname + “;” + FQDN + “/” + FullPath).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`
- Getdst 1stpath: `(Ccname + “;” + FQDN + “/” + 1stPath).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`
- Getdst FQDN: `(Ccname + “;” + FQDN).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`
- Getdst WILD: `(Ccname + “;” + WILD).MAP_STRING(“SM_CS_CONTROL”).AFTER_STR(“dst=“).BEFORE_STR(“;“)`



Conclusions

- Core Logic does nothing that is not supported / supportable
 - There is no hidden hack we need
 - We only use features and functions the NetScaler platform provides
- The Customer does not need / want to know all the intricate stuff.
- Core Logic provides a standardized method to deploy web applications
- Exceptions will be exceptions...



Core Logic - extensions

For those who can't get enough



Extensions

- TCP loadbalancing, using the SM_IP_CONTROL as whitelist/blacklist
- UDP loadbalancing
- IP-V6 / IP-V4 contentswitch combinations
- Keppath redirects
- Authorization of AAA enabled loadbalancers based on SM_VS_Control (LUA)

Improvements



- The PE_IS_LAN calculation needs improvement
- The PE_IS_SSL is not as it could be
- We'd rather do not use Policy Extensions (for authorization)

Under construction (tooling)



- Managing the tenant through NITRO
- Flow analysis of a request
- Templating applications



Thank you

Questions?