TenableAsset.json template

Template	Comments
{ "name": "Tenable Assets", "vendor_identifier": "Tenable", "comment": "Tenable assets management", "version": "3.0", "type": "REST_EVENT", "event_type": ["LEASE", "NETWORK_IPV4", "RANGE_IPV4", "FIXED_ADDRESS_IPV4", "HOST_ADDRESS_IPV4", "NETWORK_IPV6", "RANGE_IPV6", "FIXED_ADDRESS_IPV6", "HOST_ADDRESS_IPV6"], "content_type": "application/json", "headers": {"X-Requested-With": "XMLHttpRequest", "X-SecurityCenter": "${S:A:SESSID}"},	"version" must be set to "3.0" (NIOS 8.2 supports version "3.0") This template can be used with LEASE, NETWORK_IPV4, RANGE_IPV4, FIXED_ADDRESS_IPV4, HOST_ADDRESS_IPV4, NETWORK_IPV6, RANGE_IPV6, FIXED_ADDRESS_IPV6 and HOST_ADDRESS_IPV6 events/notifications. X-SecurityCenter parameter send in HTTP headers to authenticate the session
"steps": [{ "name": "DebugOnStart", "operation": "NOP", "body": "${XC:DEBUG:{H:}}${XC:DEBUG:{E:}}${XC:DEBUG:{I:}}${XC:DEBUG:{L:}}${XC:DEBUG:{S:}}${XC:DEBUG:{P:}}${XC:DEBUG:{UT:}}" },	Steps block Debug output all variables in H, E, I, L, S, O, UT name spaces

{
"name": "skip object modification",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${E:A:operation_type}", "op": "!=", "right": "INSERT"}],
"stop": true
}
},
	Stop the template execution if operation type is MODIFY or DELETE

{ "name": "assignLVars", "operation": "NOP", "body_list": ["${XC:COPY:{L:SyncDate}:{UT:TIME}}${XC:FORMAT:TRUNCATE:{L:SyncDate}:{16t}}${XC:ASSIGN:{L:Hostname}:{S:}}"] },	Assign a local variable SyncDate which will be used to populate TNBL_SyncTime extensible attribute and initialize Hostname variable

{
"name": "checkEType_Network",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${E:A:event_type}", "op": "==", "right": "LEASE"}],
"next": "checkEType_Lease"
}
	If event type is LEASE go to "checkEType_Lease" step

{
"name": "skip Sync is not requested",
"operation": "CONDITION",
"condition": {
"condition_type": "OR","statements": [{"left": "${E:A:values{extattrs}{TNBL_Sync}{value}}", "op": "==", "right": ""},{"left": "${E:A:values{extattrs}{TNBL_Sync}{value}}", "op": "==", "right": "false"}],
"stop": true}
},
	Check if it is required to sync the object (TNBL_Sync should be set to "true")

{
"name": "assignLVarsNet",
"operation": "NOP",
"body_list": [
"${XC:COPY:{L:AssetIP}:{E:values{extattrs}{TNBL_AssetIP}{value}}}",
"${XC:COPY:{L:AssetHost}:{E:values{extattrs}{TNBL_AssetHost}{value}}}",
"${XC:COPY:{L:ScanTemplate}:{E:values{extattrs}{TNBL_ScanTemplate}{value}}}",
"${XC:COPY:{L:ScanOnAdd}:{E:values{extattrs}{TNBL_ScanOnAdd}{value}}}",
"${XC:COPY:{L:Obj_ref}:{E:values{_ref}}}",
"${XC:ASSIGN:{L:SaveEA}:{S:true}}"
]},{
"name": "Set_IPF_Network",
"operation": "CONDITION",
"condition": {
"condition_type": "OR",
"statements": [{"left": "${E:A:event_type}", "op": "==", "right": "NETWORK_IPV4"}, {"left": "${E:A:event_type}", "op": "==", "right": "NETWORK_IPV6"}],
"eval": "${XC:COPY:{L:Network}:{E:values{network}}}${XC:COPY:{L:IPObject}:{E:values{network}}}${XC:ASSIGN:{L:ObjType}:{S:NET}}"}
},{
"name": "Set_IPF_Range",
"operation": "CONDITION",
"condition": {
"condition_type": "OR",
"statements": [{"left": "${E:A:event_type}", "op": "==", "right": "RANGE_IPV4"},{"left": "${E:A:event_type}", "op": "==", "right": "RANGE_IPV6"}],
	Assign the local variables from extensible attributes and event variables.
	
	If TNBL_AssetIPID, TNBL_AssetHostID, TNBL_ScanTemplateID are not defined, set them to the default value ("0")

```
"eval": "${XC:ASSIGN:{L:ObjType}:{S:RANGE}}",
"else_next": "Set_IPF_Host_IPv4"}
},{
"name": "Create a range",
"operation": "SERIALIZE",
"serializations": [{"destination": "L:IPObject","content":
"${E:A:values{start_addr}}-${E:A:values{end_addr}}"}]
},{
"name": "Set_IPF_Host_IPv4",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left": "${E:A:event_type}", "op":
"==", "right": "HOST_ADDRESS_IPV4"}],
"eval":
"${XC:COPY:{L:IPObject}:{E:values{ipv4addr}}}${XC:COPY:{L:Hostnam
e}:{E:values{host}}}${XC:ASSIGN:{L:IPv}:{S:ipv4addr}}${XC:ASSIGN:{L:
ObjType}:{S:HOST}}"}
},{
"name": "Set_IPF_Host_IPv6",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left": "${E:A:event_type}", "op":
"==", "right": "HOST_ADDRESS_IPV6"}],
"eval":
"${XC:COPY:{L:IPObject}:{E:values{ipv6addr}}}${XC:COPY:{L:Hostnam
e}:{E:values{host}}}${XC:ASSIGN:{L:IPv}:{S:ipv6addr}}${XC:ASSIGN:{L:
ObjType}:{S:HOST}}"}
},{
"name": "Set_IPF_Fixed_IPv4",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left": "${E:A:event_type}", "op":
"==", "right": "FIXED_ADDRESS_IPV4"}],
"eval":
"${XC:COPY:{L:IPObject}:{E:values{ipv4addr}}}${XC:ASSIGN:{L:ObjTyp
e}:{S:FIXEDIP}}"}
},{
"name": "Set_IPF_Fixed_IPv6",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left": "${E:A:event_type}", "op":
"==", "right": "FIXED_ADDRESS_IPV6"}],
"eval":
"${XC:COPY:{L:IPObject}:{E:values{ipv6addr}}}${XC:ASSIGN:{L:ObjTyp
e}:{S:FIXEDIP}}"}
},{
"name": "Set_NetToSite",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left":
"${E:A:values{extattrs}{TNBL_AddNet}{value}}", "op": "==", "right": ""}],
"eval": "${XC:ASSIGN:{L:NetToSite}:{S:false}}",
```

"else_eval":
"${XC:COPY:{L:NetToSite}:{E:values{extattrs}{TNBL_AddNet}{value}}}"}
},{
"name": "Set_RangeToSite",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left":
"${E:A:values{extattrs}{TNBL_AddRange}{value}}", "op": "==", "right":
""}],
"eval": "${XC:ASSIGN:{L:RangeToSite}:{S:false}}",
"else_eval":
"${XC:COPY:{L:RangeToSite}:{E:values{extattrs}{TNBL_AddRange}{val
ue}}}"}
},{
"name": "Set_AddByHostname",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left":
"${E:A:values{extattrs}{TNBL_AddByHostname}{value}}", "op": "==",
"right": ""}],
"eval": "${XC:ASSIGN:{L:AddByHostname}:{S:false}}",
"else_eval":
"${XC:COPY:{L:AddByHostname}:{E:values{extattrs}{TNBL_AddByHost
name}{value}}}"}
},	

{
"name": "findRef_Host",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${L:A:ObjType}","op":
"!=","right": "HOST"}],
"next": "Fin_Vars_Init"}
},{
"name": "findRef_Host_ch_Delete",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${E:A:operation_type}",
"op": "==", "right": "DELETE"}],
"next": "Fin_Vars_Init"}
},{
"name": "Get Host _ref",
"operation": "GET",
"transport": {"path":
"record:host?_return_fields=name,extattrs&network_view=${E:A:values{
network_view}}&name=${L:U:Hostname}&${L:U:IPv}=${L:U:IPObject}"},
"wapi": "v2.7"
},{
"operation": "CONDITION",
"name": "wapi_response_get_ref",
"condition": {
	Looking for _ref property for HOST object. For other objects (except LEASE) go to "Fin_Vars_Init" step

"condition_type": "AND", "statements": [{"left": "${P:A:PARSE[0]{_ref}}", "op": "==", "right": ""}], "error": true, "else_eval": "${XC:COPY:{L:Obj_ref}:{P:PARSE[0]{_ref}}}"} },{ "name": "Debug P vars", "operation": "NOP", "body": "${XC:DEBUG:{P:}}" },	

{
"name": "checkEType_Lease",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${E:A:event_type}", "op": "!=","right": "LEASE"}],
"next": "Fin_Vars_Init"}
},
	For all objects (except LEASE) go to "Fin_Vars_Init" step (actually on this step only HOST object is expected).

{
"name": "findRef_Host_ch_Delete",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${E:A:operation_type}", "op": "==", "right": "DELETE"}],
"next": "Fin_Vars_Init"}
},
	If operation_type is DELETE go to "Fin_Vars_Init" step

{
"name": "skip if not defined for lease",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left": "${E:A:ip.extattrs{TNBL_Sync}}", "op": "==", "right": ""},{"left": "${E:A:ip.extattrs{TNBL_Sync}}", "op": "==", "right": "false"}],
"stop": true}
},
	Stop the template execution if LEASE objects sync is not requested

{
"name": "assignLVarsLease",
"operation": "NOP",
"body_list": [
"${XC:COPY:{L:Network}:{E:values{network}}}",
"${XC:COPY:{L:IPObject}:{E:values{address}}}",
"${XC:COPY:{L:AssetIP}:{E:ip.extattrs{TNBL_AssetIP}}}",
"${XC:COPY:{L:AssetHost}:{E:ip.extattrs{TNBL_AssetHost}}}",
"${XC:COPY:{L:Sync}:{E:ip.extattrs{TNBL_Sync}}}",
"${XC:COPY:{L:ScanTemplate}:{E:ip.extattrs{TNBL_ScanTemplate}}}",
"${XC:COPY:{L:ScanOnAdd}:{E:ip.extattrs{TNBL_ScanOnAdd}}}",
"${XC:COPY:{L:Hostname}:{E:values{client_hostname}}}",
"${XC:ASSIGN:{L:SaveEA}:{S:false}}",
"${XC:ASSIGN:{L:ObjType}:{S:LEASE}}"]
},{
	Assign local variables for the LEASE event type

"name": "Set_L_SiteIPID",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left":
"${E:A:ip.extattrs{TNBL_AssetIPID}}", "op": "==", "right": ""}],
"eval": "${XC:ASSIGN:{L:AssetIPID}:{I:0}}",
"else_eval":
"${XC:COPY:{L:AssetIPID}:{E:ip.extattrs{TNBL_AssetIPID}}}"}
},{
"name": "Set_L_SiteHostID",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left":
"${E:A:ip.extattrs{TNBL_AssetHostID}}", "op": "==", "right": ""}],
"eval": "${XC:ASSIGN:{L:AssetHostID}:{I:0}}",
"else_eval":
"${XC:COPY:{L:AssetHostID}:{E:ip.extattrs{TNBL_AssetHostID}}}"}
},{
"name": "Set_L_ScanTemplateID",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [{"left":
"${E:A:ip.extattrs{TNBL_ScanTemplateID}}", "op": "==", "right": ""}],
"eval": "${XC:ASSIGN:{L:ScanTemplateID}:{S:false}}",
"else_eval":
"${XC:COPY:{L:ScanTemplateID}:{E:ip.extattrs{TNBL_ScanTemplateID}
}}"}
},	

{
"name": "Fin_Vars_Init",
"operation": "NOP",
"body": "${XC:DEBUG:{L:}}"
},{
"name": "Check AssetIPID",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${L:A:AssetIPID}", "op":
"!=", "right": "0"},{"left": "${L:A:ObjType}", "op": "!=", "right": "HOST"}],
"next": "SelectAssetID"}
},{
"name": "Check AssetHostID",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${L:A:AssetHostID}",
"op": "!=", "right": "0"},{"left": "${L:A:ObjType}", "op": "==", "right":
"HOST"},{"left": "${L:A:AddByHostname}", "op": "==", "right": "true"}],
"next": "SelectAssetID"}
},{
"name": "Check AssetHostIDbyIP",
"operation": "CONDITION",
"condition": {
	If AssetIPID, AssetHostID were set (or inherited) we do not need to find these IDs and should go to "SelectAssetID" step.

"condition_type": "AND", "statements": [{"left": "${L:A:AssetIPID}", "op": "!=", "right": "0"},{"left": "${L:A:ObjType}", "op": "==", "right": "HOST"},{"left": "${L:A:AddByHostname}", "op": "==", "right": "false"}], "next": "SelectAssetID"} },	

{
"name": "Request all assets",
"parse": "JSON",
"operation": "GET",
"transport": {"path": "/asset"}
},{
"name": "Check all assets request on errors",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op": "!=", "right": "0"}],
"else_eval": "${XC:COPY:{L:object_list}:{P:response{manageable}}",
"error": true}
},
	Requests all assets from Tenable SC and store manageable assets in a local variable

{
"name": "Check if asset list is empty",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:L:object_list}", "op": "==", "right": "0"}],
"stop": true}
},{
"name": "Pop asset from the list",
"operation": "VARIABLEOP",
"variable_ops": [
{
"operation": "POP",
"type": "DICTIONARY",
"destination": "L:an_object",
"source": "L:object_list"}]
},{
"name": "check an asset AssetIP",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:AssetIP}", "op": "!=", "right": "${L:A:an_object{name}}"}],
"else_eval": "${XC:COPY:{L:AssetIPID}:{L:an_object{id}}}"}
},{
"name": "check an asset AssetHost",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:AssetHost}", "op": "!=", "right": "${L:A:an_object{name}}"}],
"else_eval": "${XC:COPY:{L:AssetHostID}:{L:an_object{id}}}"}
},{
	The loop is go through the list of manageable assets and tries to find IDs for the lists defined in TNBL_AssetIP and TNBL_AssetHosts

"name": "check AssetIPID & AssetHostID are set", "operation": "CONDITION", "condition": { "condition_type": "OR","statements": [{"left": "${L:A:AssetIPID}", "op": "==", "right": "0"},{"left": "${L:A:AssetHostID}", "op": "==", "right": "0"}], "next": "Check if asset list is empty"} },	

{
"name": "Check Net n Range",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:ObjType}", "op": "!=", "right": "NET"},{"left": "${L:A:ObjType}", "op": "!=", "right": "RANGE"}],
"next": "SelectAssetID"}
},
	If the object is not NETWORK or RANGE go to "SelectAssetID" step

{ "name": "Update AssetsID for Net n Range", "operation": "PUT", "transport": {"path": "${L:A:Obj_ref}"}, "wapi": "v2.7", "wapi_quoting": "JSON", "body_list": ["{\"extattrs+\":{\"TNBL_AssetIPID\": { \"value\": \"${L:A:AssetIPID}\"},\"TNBL_AssetHostID\": { \"value\": \"${L:A:AssetHostID}\"}}}"] },	Update TNBL_AssetIPID, TNBL_AssetHostID extensible attributes for networks and ranges only

{
"name": "Check NetToSite exit",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:NetToSite}", "op": "!=", "right": "true"},{"left": "${L:A:ObjType}", "op": "==", "right": "NET"}],
"stop": true}
},{
"name": "Check RangeToSite exit",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:RangeToSite}", "op": "!=", "right": "true"},{"left": "${L:A:ObjType}", "op": "==", "right": "RANGE"}],
"stop": true}
},
	If it is not required to sync a network or a range with Tenable, stop the template

{
"name": "SelectAssetID",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${L:A:AssetHostID}", "op": "!=", "right": "0"},{"left": "${L:A:Hostname}", "op": "!=", "right":
	Selecting which asset list (TNBL_AssetIP or TNBL_AssetHost) to use

""},{"left": "${L:A:ObjType}", "op": "==", "right": "HOST"},{"left": "${L:A:AddByHostname}", "op": "==", "right": "true"}], "eval": "${XC:ASSIGN:{L:UpdateObject}:{S:definedDNSNames}}${XC:COPY:{L:AssetID}:{L:AssetHostID}}${XC:COPY:{L:SyncObject}:{L:Hostname}}", "else_eval": "${XC:ASSIGN:{L:UpdateObject}:{S:definedIPs}}${XC:COPY:{L:AssetID}:{L:AssetIPID}}${XC:COPY:{L:SyncObject}:{L:IPObject}}"} },	

{
"name": "GetAssets",
"operation": "GET",
"parse": "JSON",
"transport": {"path": "/asset/${L:A:AssetID}"}
},{
"name": "Check GetAssets",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op": "!=", "right": "0"}],
"error": true}
},{
"name": "GetCurAssets",
"operation": "CONDITION",
"condition": {
"condition_type": "AND", "statements": [{"left": "${L:A:UpdateObject}", "op": "==", "right": "definedDNSNames"}],
"eval":
"${XC:COPY:{L:CurAssets}:{P:response{typeFields}{definedDNSNames}}}",
"else_eval":
"${XC:COPY:{L:CurAssets}:{P:response{typeFields}{definedIPs}}}"}
},
	Retrieving the current asset list configuration and copying it to a local variable

{
"name": "Update assets",
"operation": "PATCH",
"parse": "JSON",
"headers": {"X-Requested-With": "XMLHttpRequest", "X-SecurityCenter": "${S:A:SESSID}"},
"transport": {"path": "/asset/${L:A:AssetID}"},
"body_list": [
"{\"${L:A:UpdateObject}\":\"${L:A:CurAssets},${L:A:SyncObject}\",\"description\":\"Updated by Infoblox Outbound API at ${L:A:SyncDate}\"}"]
},{
"name": "Check assets update",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op": "!=", "right": "0"}],
"error": true}
	Updating Tenable SC asset list

},	

{
"name": "checkSaveAssetsID",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:SaveEA}", "op":
"!=", "right": "true"}],
"next": "Check if ScanOnAdd"
}},{
"name": "Update AssetsID",
"operation": "PUT",
"transport": {"path": "${L:A:Obj_ref}"},
"wapi": "v2.7",
"wapi_quoting": "JSON",
"body_list": [
"{\"extattrs+\":{\"TNBL_AssetIPID\": { \"value\":
\"${L:A:AssetIPID}\"},\"TNBL_AssetHostID\": { \"value\":
\"${L:A:AssetHostID}\"},\"TNBL_SyncTime\": { \"value\":
\"${L:A:SyncDate}\"}}}"]
},
	Updating extensible attributes if it is possible (e.g. we can not update LEASE object)

{
"name": "Check if ScanOnAdd",
"operation": "CONDITION",
"condition": {
"condition_type": "OR","statements": [{"left": "${L:A:ScanOnAdd}", "op":
"!=", "right": "true"},{"left": "${L:A:ScanTemplate}", "op": "==", "right":
""},{"left": "${L:A:ObjType}", "op": "==", "right": "NET"},{"left":
"${L:A:ObjType}", "op": "==", "right": "RANGE"}],
"next": "Fin"}
},
	If Scan on Add was not requested go to "Fin"

{ "name": "assignScanVars", "operation": "NOP", "body_list": ["${XC:COPY:{L:ScanDate}:{UT:TIME}}${XC:FORMAT:TRUNCATE:{L:S canDate}:{10t}}", "${XC:COPY:{L:ScanSchTime}:{UT:EPOCH}}${XC:FORMAT:DATE_ST RFTIME:{L:ScanSchTime}:{%Y%m%dT%H%M59000Z}}"] },	Assign local scan variables

{
"name": "Get a UserID",
"operation": "GET",
"parse": "JSON",
"transport": {"path": "/currentUser"}
},{
"name": "Check a user",
"operation": "CONDITION",
"condition": {
	Get Tenable Security Center user ID.

"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op": "!=", "right": "0"}], "error": true, "else_eval": "${XC:COPY:{L:TNBL_UserId}:{P:response{id}}}"} },	

{
"name": "checkIfExistsScanTemplateID",
"operation": "CONDITION",
"condition": {
"condition_type": "OR","statements": [{"left": "${L:A:ScanTemplateID}",
"op": "!=", "right": "false"}],
"next": "Copy a scan template"}
},
	Check if TNBL_ScanTemplateID was already determined or inherited

{
"name": "Request all scans",
"parse": "JSON",
"operation": "GET",
"transport": {"path": "/scan"}
},{
"name": "Check all scans request on errors",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op": "!=", "right": "0"}],
"else_eval": "${XC:COPY:{L:object_list}:{P:response{manageable}}",
"error": true}
},
	Request all active scans and save manageable scans into a local variable

{
"name": "Check if list is empty",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:L:object_list}", "op": "==", "right": "0"}],
"stop": true}
},{
"name": "Pop object from the list",
"operation": "VARIABLEOP",
"variable_ops": [{
"operation": "POP",
"type": "DICTIONARY",
"destination": "L:an_object",
"source": "L:object_list"}]
},{
"name": "check an object",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:ScanTemplate}",
"op": "!=", "right": "${L:A:an_object{name}}"}],
"next": "Check if list is empty",
	The loop is go through the list of manageable assets and tries to find IDs for the active scan.

"else_eval": "${XC:COPY:{L:ScanTemplateID}:{L:an_object{id}}}"} },	

{
"name": "checkSaveScanID",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${L:A:SaveEA}", "op":
"!=", "right": "true"}],
"next": "Copy a scan template"}
},{
"name": "Update ScanID",
"operation": "PUT",
"transport": {"path": "${L:A:Obj_ref}"},
"wapi": "v2.7",
"wapi_quoting": "JSON",
"body_list": ["{\"extattrs+\":{\"TNBL_ScanTemplateID\": { \"value\":
\"${L:A:ScanTemplateID}\"}}}"]
},
	Update TNBL_ScanTemplateID extensible attribute if it is possible (e.g. we can not update LEASE object)

{
"name": "Copy a scan template",
"operation": "POST",
"parse": "JSON",
"transport": {"path": "/scan/${L:A:ScanTemplateID}/copy"},
"body_list": [
"{\"targetUser\":{\"id\":\"${L:A:TNBL_UserId}\"},\"name\":\"${L:A:SyncObj
ect} scan requested by IB Outbound API by the asset creation at
${L:A:ScanSchTime}\"}"]
},{
"name": "Check Copy",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op":
"!=", "right": "0"}],
"error": true}
},
	Copy an active scan template and populate the name field

{
"name": "Run a scan",
"operation": "PATCH",
"parse": "JSON",
"transport": {"path": "/scan/${P:A:response{scan}{id}}"},
"body_list": [
"{\"ipList\":\"${L:A:SyncObject}\",\"schedule\":{\"repeatRule\":
\"FREQ=NOW;INTERVAL=1\", \"type\": \"now\"}}"]
},{
"name": "Check Run a scan",
"operation": "CONDITION",
"condition": {
"condition_type": "AND","statements": [{"left": "${P:A:error_code}", "op":
"!=", "right": "0"}],
	Execute the template which was copied on the previous step

"error": true} },	

{
"name": "checkSaveLastScan",
"operation": "CONDITION",
"condition": {
"condition_type": "OR", "statements": [
{"left": "${L:A:SaveEA}", "op": "!=", "right": "true"},
{"left": "${L:A:EASource}", "op": "==", "right": "Net"}],
"next": "Fin"}
},{
"name": "Update_LastScan",
"operation": "PUT",
"transport": {"path": "${L:A:Obj_ref}"},
"wapi": "v2.7",
"wapi_quoting": "JSON",
"body_list": [
"{\"extattrs+\":{\"TNBL_ScanTime\": { \"value\": \"${L:U:ScanDate}\"}}}"]
},
	Update TNBL_ScanTime extensible attribute if it possible

{
"name": "Fin",
"operation": "NOP",
"body": "${XC:DEBUG:{L:}}${XC:DEBUG:{E:}}${XC:DEBUG:{P:}}"
}]
}
	Debug output of the variables