PATH	HTTP VERB	PATH PARAM	BODY	FUNCTIONALITY
/version	GET	N/A	N/A	return PBCONF version number
/version/{namespace}	GET	“node”, “device”, etc	N/A	return version number of namespace
/namespace	GET	N/A	N/A	return version, device, node, reports, policy and any other dynamically added namespace
/device	HEAD	N/A	N/A	return “Last-Modified” tag in header
	GET	N/A	N/A	return Device list (Name, Id, Parent Node Id)
	POST	N/A	Device	return StatusCreated, add new device to node, pass notification upstream
	PATCH	N/A	Device	return StatusOk, update device if this node is the owner, else pass to owner node to update
/device/{devid}	GET	deviceId=int	N/A	return device
	PATCH	deviceId=int	Device	return StatusOk, update device if this node is the owner, else pass to owner node to update
	DELETE	deviceId=int	N/A	return StatusOk, delete device if this node is the owner, else pass to owner node to delete
/device/{devid}/config	GET	deviceId=int	N/A	return device configuration information from repository
	PATCH	deviceId=int	change.ChangeData	return StatusOk, update device config if this node is the owner, else pass to owner node to update
/device/{devid}/meta	GET	deviceId=int	N/A	return StatusOK, device meta-data list
	PATCH	deviceId=int	key-value pair	return StatusOK, update device metadata if this node is the owner, else pass to owner node to update
/device/{devid}/{cfgkey}	GET	deviceId=int,cfgkey=string	N/A	return key-value pair
	DELETE	deviceId=int,cfgkey=string	N/A	return StatusOk, delete device config key-value pair if this node is the owner, else pass to owner node to delete
/node	HEAD	N/A	N/A	return “Last-Modified” tag in header
	GET	N/A	N/A	return node list (name, id)
	POST	N/A	Node	return StatusCreated, create new node.
	PATCH	N/A	Node	return StatusOk, updates node name or config items
/node/{nodeid}	GET	nodeId=int	N/A	return node
	PATCH	nodeId=int	node	return StatusOk, updates node name or config items
	DELETE	nodeId=int	N/A	return statusOk, delete node
/node/{nodeid}/{cfgkey}	GET	nodeId=int,cfgkey=string	N/A	return key-value pair
	DELETE	nodeId=int,cfgkey=string	N/A	delete node cfg key-value pair
/policy	GET	N/A	N/A	return list of policy names
	PATCH	N/A	policy	If this node i then save to otherwise pass it up
	HEAD	N/A	N/A	return "X-Pbconf-Policy-LastCommitId" tag in header
/policy/all	GET	N/A	N/A	return list of policies (with data), is gzipped
/policy/default/{nodename}	HEAD	nodename=string	N/A	called by downstream node heartbeat and marked as checked in. If this nodename doesn't exist, it is added to this nodes list of downstream nodes.
/policy/{policyname}	GET	policyname=string	N/A	return policy content from repo
	PATCH	policyname=string	policy	If this node is master, validate against ontology, then save to repo, otherwise pass it up
	DELETE	policyname=string	N/A	If this node is master, delete the policy. Otherwise send it up to master
/policy/{devid}/validate	????			
/policy/{devid}/validate/{valid}	???			
/reports	GET	N/A	N/A	return list of report names that exist in the repo
	POST	N/A	PbReportQueryHttp	return Status MAY NEED MORE WORK
	HEAD	N/A	N/A	return "X-Pbconf-Report-LastCommitId" tag in header
/reports/{reportid}	GET	reportid=string	N/A	return report query
	PUT	reportid=string	PbReportQueryHttp	update/add r report run or timer set up to run report periodically
	DELETE	reportid=string	N/A	delete report from repo
/reports/report/{reportid}	GET	reportid=string	N/A	return results of running the report query

In addition to the above routes, all routes will work if prepended by a version. For e.g /vXX/reports/{reportid}, where XX is the int from 1 to whatever the latest version for that namespace is currently.