Nive Web-API Chart

Version 0.9.1

Identity Management (userstore)	Return values	Error codes	
.token({identity, password})	{token, *message}	422, 401	Get a security token for service requests..
.signin({identity, password})	{result, *message}	422, 401	Handles the token automatically as cookie.
.signout()	{result}		Resets the security cookie.
.identity()	{name, email, id, realname}		
.name()	{identity, realname}		
.profile()	{name, email, realname, data, notify, activity, id}		
.authenticated({*groups})	{result}	422	groups = One or more groups
.allowed({permission})	{result, {permission: true/false}, *message, *invalid}	422	Test if current user is allowed to acces the fuction. Supports single and multiple permission names.
.signupDirect({name, email, password, data})	{result, *message, *invalid}	413, 422	New accounts are instantly activated.
.signupOptin({name, email, password, data})	{result, *message, *invalid}	413, 422	The account is activated by verifying the users email.
.signupReview({name, email, password, data})	{result, *message, *invalid}	413, 422	The account has to be activated by the administrator and the users email is verified.
.signupSendpw({name, email, data})	{result, *message, *invalid}	413, 422	The email address is verified by sending a generated password .
.signupUid({email, password, data})	{result, *message, *invalid}	413, 422	This function generates a 30 byte long unique identifier for the user.
.activate({token})	{result, *message}	422	Follow up for `signupOptin()` or `review()` calls.
.update({data, realname, notify})	{result, *message, *invalid}	422	
.updatePassword({password, newpassword})	{result, *message, *invalid}	422	
.updateEmail({email})	{result, *message, *invalid}	422	The new email. Verification may be required.
.verifyEmail({email})	{result, *message, *invalid}	422	
.verifyEmail2({token})	{result, *message}	422	`token` is generated by `verifyEmail()`
.resetPassword({identity})	{result, *message}	422	Identity depends on backend settings.
.resetPassword2({token, newpassword})	{result, *message, *invalid}	422	`token` is generated by `resetPassword()`
.message({message})	{result, *message}	422	Sends a message to the domains user admin
.disable()	{result}		Deactivate a users account.
.delete()	{result}		Delete a users account. Disables and sets deleted state. Use removeUser() to completely remove a users account.
.review({identity, action})	{result, *message}	422	Accept or reject a pending sign up request. Action is either 'accept', `optin`,'reject', `activate` or `disable`. If 'optin' is true a activation mail is triggered. See 'signupOptin()'. If action='accept' or `reject` the users state is only changed if 'user.pending' is true.

.getUser({identity})	{name, email, realname, notify, data, groups, activity, reference, active, pending, token}	422	Administration command.
.setUser({identity, values})	{result, *message, *invalid}	422	Administration command. See getUser().
.removeUser({identity})	{result, *message}	422	Administration command.
.list({active, pending, sort, order, size, start})	{[users], start, size, *message, *invalid}	422	Administration command. Each returned user contains (id, name, email, realname, pending, active, activity)
.identities({active, pending, order, size, start})	{[users], start, size, *message, *invalid}	422	Administration command. Lists user identites as string only (id, name/email).
.getPermissions()	[{permission, groups}]		Retireve local access permissions.
.setPermissions([permissions])	{result, *message}	422	Set the items local access permissions {permission, groups [, action]}. Pass action=revoke to remove a permission.
.ping()	{result}		Health check. Returns status='OK'.
Data Storage (datastore)			
.getItem({key, *owner}) .getItem({id})	{items, *message}	404, 422	Pass a list to read a batch of items.
.newItem({key, value, *owner}) .newItem([items])	{result: count stored, success: [position,key,id,owner,timestamp], *message, *invalid}	413, 422	'items' = Pass multiple items as a list.
.setItem({key, value, *owner, *id}) .setItem([items])	{result: count stored, success: [position,key,id,owner,timestamp], *message, *invalid}	404, 422	'items' = Pass multiple items as a list. 'key' is always required even If passing the items 'id'.
.removeItem({key, *owner}) .removeItem({id}) .removeItem([items])	{result: count deleted, success: [position,key,id,owner], message}	422	'items' = Pass multiple items as a list.
.list({key, sort, order, size, start, owner})	{items, start, size, *message, *invalid}	422	'key' = pass multiple keys as list; 'sort' either key, value or timestamp; 'order' either < or >; owner = 'sys:all' to match all users.
.keys({order, size, start, owner})	{keys, start, size, *message, *invalid}	422	'sort' either key, value or timestamp; 'order' either < or >; owner = 'sys:all' to match all users.
.allowed({permission})	{result, {permission: true/false}, *message, *invalid}	422	Test if current user is allowed to acces the fuction. Supports single and multiple permission names.
.getPermissions()	[{permission, groups}]		Retireve the items local access permissions.
.setPermissions([permissions])	{result, *message}	422	Set the items local access permissions {permission, groups [, action]}. Pass action=revoke to remove a permission.
.getOwner({key}) .getOwner({id})	{items, *message}	404, 422	Pass a list to read a batch of items. Each returned itesm includes {key, id, owner}
.setOwner({newOwner, key, owner}) .setOwner({newOwner, id}) .setOwner({newOwner, [items]})	{result: count stored, success: [position,key,id,owner,timestamp], *message}	422	Set the items owner. items = one or multiple {key/id, owner} objects
.ping()	{result}		Health check. Returns status='OK'. Root only.

File Storage (filestore)			
.getItem({path})	{name, type, size, mime, header, ctime, mtime, cache}	404	Returns the files/folders meta data.
.newItem({path, name, contents, type, mime, header, decode})	{result, *message, *invalid}	404, 409, 413, 422	Adds a new file or folder.
.setItem({path, contents, mime, header, decode})	{result, *message, *invalid}	404, 413, 422	Update a file or folder.
.removeItem({path, recursive})	{result: count deleted, *message}	404	Delete a file or folder. Set recursive=true to delete non empty folders..
.read({path})	<File data>	404	Read the files contents.
.write({path, contents, mime})	{result, *message}	404, 413, 422	Streaming file writer.
.move({path, newpath})	{result, *message}	404, 413	Move a file or folder to newpath.
.list({path, type, sort, order, size, start})	{[items], start, size}	404	List items contained in a folder. Each item contains {name, type, size, mime, mtime, ctime}
.allowed({path, permission})	{result, {permission: true/false}, *message, *invalid}	404	Test if current user is allowed to acces the fuction. Supports single and multiple names.
.getPermissions({path})	[{permission, groups}]	404, 413	Retireve the items local access permissions.
.setPermissions({path, permissions})	{result, *message}	404, 413	Set the items local access permissions {permission, groups[, action]}. Pass action=revoke to remove a local permission.
.getOwner({path})	{owner}	404, 413	Retireve the items owner.
.setOwner({path, owner})	{result, *message}	404, 413	Set the items owner.
.ping()	{state}		Health check. Returns status='OK'. Root only.
Endpoint Url (endpoint)			
.makeUrl({method, service, domain, path, secure, version}, extendedPath)	url		Constructs a api endpoint url for a specific method..

* optional parameter or response value.

Error codes: http reponse status codes pointing to the reason for the failure. Success has always status 200.