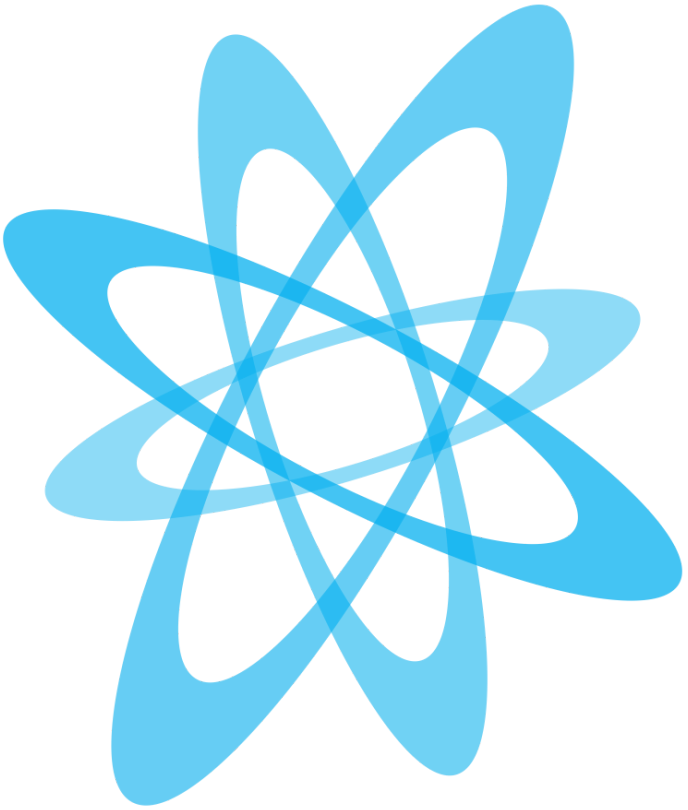
Sislander

API



Documentación

Índice

Introducción a la API de Sislander

La API de Sislander permite que otras aplicaciones se comuniquen con el sistema de control del servidor y obtengan información del mismo o le den órdenes.

Las aplicaciones pueden conectar con la API de Sislander incluida en cada sistema Sislander con versión 21.03 o superior a través de varios protocolos (http y https) y varios métodos (POST o GET). Pueden enviar las consultas en texto simple o encriptado y utilizar dos diferentes sintaxis para las consultas: uno de estructura similar a GraphQL (ya disponible) y otro que emulará la pseudo-API-REST de mikrotik (disponible más adelante).

En este documento nos enfocaremos en la combinación de protocolo, método y sintaxis de consulta más sencilla, segura y consolidada a la fecha.

CONEXIÓN

En el cliente API, o sea la aplicación que consultará la API de Sislander, recomendamos utilizar:

- Protocolo HTTPS
- Autenticación por headers, auth o post
- Método POST

Protocolo HTTPS

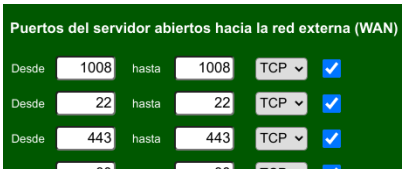
- La URL debe contener protocolo, IP y la ruta ***/api/***
  Ejemplos:
  a. https: https://186.42.130.179/api/
  (IP pública, generalmente en la WAN)
  b. https: https://10.42.42.1/api/
  (IP privada, dentro de la red)

- Cuando se utiliza el protocolo HTTPS se debe verificar un certificado, como hacen los navegadores en la mayoría de los sitios web.
  Hay 2 opciones para que la aplicación cumpla este requisito del protocolo:

  a. Configurar la aplicación cliente para que omita la verificación o ignore errores de certificado (si el cliente tiene esta opción disponible)

  b. Cargar en la aplicación cliente el certificado para cada servidor que se consulte. El certificado lo puedes copiar en el sistema de control del sistema Sislander a consultar, en la página *Operadores*, haciendo clic en la ayuda contextual que aparece junto a la opción *Acceso por API* .

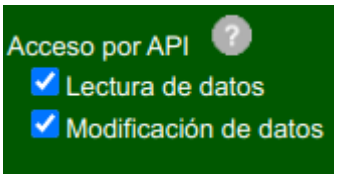
- ¡Atención! Es importante verificar que el puerto 443 se encuentre abierto en la/s redes WAN si se va a conectar a la API por algunas de las IP WAN. Puedes verificar y agregar si es necesario la apertura del puerto 443 en *Redes >> Avanzadas* en cada una de las WAN.



Puertos abiertos en una Red WAN, incluyendo el 443

Autenticación

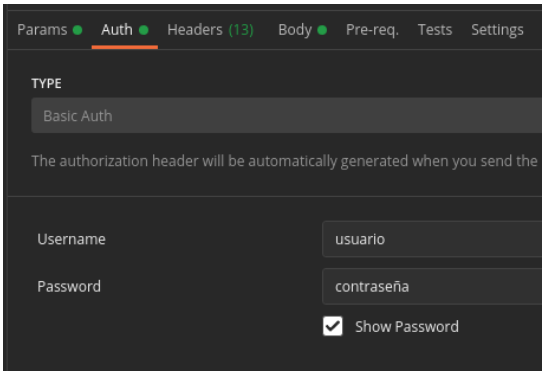
- En el sistema Sislander al cual quieres conectar se debe configurar al menos un *operador* con privilegios de *Acceso por API*. Estas serán las credenciales que debe enviar la aplicación que consultará a la API de Sislander junto con cada consulta. Por el momento la API no distingue entre privilegios de lectura y de escritura pero lo hará más adelante. Por ahora, marcar cualquiera de estos o ambos sirve para todas las consultas.



Operadores en sistema de control de Sislander

- Desde la aplicación que consultará a la API de Sislander, debes enviar las credenciales junto con **cada consulta**, con **uno cualquiera** de estos métodos:

  a. Autenticación tipo Basic Auth



Ejemplo en Postman

  b. Headers *AUTH_USER* y *AUTH_PW*

Ejemplo en Postman

c. POST requests con nombre *AUTH_USER* y *AUTH_PW*

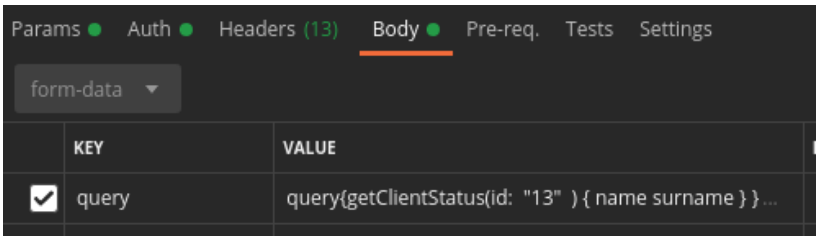
Ejemplo en Postman

- En futuras versiones la API de SIslander retornará también un token que se podrá utilizar en lugar de las credenciales luego de la primer autenticación para ahorrar transferencia y acelerar el tiempo de respuesta

Método POST

Requests aceptados por la API de Sislander:

- *query*
  El único request indispensable ya que contiene la consulta propiamente dicha en el formato JSON/GraphQL con la sintaxis aceptada por Sislander API (ver siguiente sección SINTAXIS)



Ejemplo en Postman

SINTAXIS

El valor del *query* debe ser una consulta aceptada por Sislander API de acuerdo a las funciones que están disponibles (ver siguiente sección FUNCIONES).

La sintaxis es similar a la de GraphQL y a medida que evolucione se hará más semejante a este lenguaje de consultas, integrando sus potencialidades. A su vez, el formato de las consultas tipo GraphQL es en gran medida el conocido JSON.

Sislander retornará la respuesta en formato JSON en el cuerpo mismo de la transferencia http.

Formato de las consultas a la API

```
query {
      getClientStatus(id:"13") { }
}
```

- Se encabeza con la palabra clave *query* para emular a GraphQL y luego se abren las llaves que engloban las consultas propiamente dichas. Las queries que realizan cambios en los datos se pueden encabezar con *mutation* en lugar de *query* como en GraphQL; sin embargo, *query* se puede utilizar en todos los casos.

  Por ejemplo, son igualmente válidas

  - ```
    query{
          delClient(id:"777"){data result}
    }
    ```
  - ```
    mutation{
          delClient(id:"777"){data result}
    }
    ```

- Dentro de las llaves se pueden escribir 1 o varias consultas -las queries propiamente dichas- utilizando las funciones disponibles en la API de Sislander.

  Se recomienda en todas las funciones utilizar el siguiente formato:

  función(parámetros separados por comas o saltos de línea) {
  campos a retornar separados por espacios o saltos de línea
  }

Ejemplos:

```
query{
	getClient(id:"17", plan:"2"){ id name firstName lastName }
}
```

```
query{
	getClient(
		id:"17"
		plan:"2"
	){
		id
		name
		firstName
		lastName
	}
}
```

función

Uno o varios llamados a funciones de la API de Sislander (ver siguiente sección *FUNCIONES*).

```
query{
  delClient(id:"56"){}
}
```

Si se ejecuta varias veces la misma función en un mismo query, se le agrega un prefijo variable cualquiera (un *alias*).

```
query{
  client1:delClient(id:"56"){}
  client2:delClient(id:"128"){}
}
```

Pueden consultarse funciones distintas en un mismo query (agregando un alias cuando la función se utiliza más de una vez como se menciona en el párrafo anterior).

```
query{
	c1:getClient(id:"443"){ id name firstName lastName }
	c2:getClient(id:"911"){ id name phone email }
	getClientDevices(id:"911"){ id connection byLogin byDate}
}
```

parámetros

Claves y valores para filtrar la consulta.

- Cada parámetro debe tener el formato
  clave:"valor"
  * Clave y valor separados por **:** (2 puntos)
  * Con o sin espacio después de los 2 puntos
  * Valor **siempre** entre comillas excepto si es *true* o *false* en cuyo caso **no** deben utilizarse comillas

Ejemplo:

```
query{
	editClient(
		id:"9"
		name:"Miguel de Cervantes Saavedra"
		phone:"91-9546-1254"
		plan:"7"
		byLogin:true
	){
		id
		name
		phone
		plan
	}
}
```

- Cada parámetro (par de clave:valor) se separa del siguiente con un salto de línea o con una coma (no hace falta coma después del último parámetro)

- Si no hay parámetros se dejan sólo los paréntesis ( )

campos

Campos con información que se espera retorne el query.

- Los campos de retorno son opcionales. Se pueden indicar todos, algunos o ninguno. Algunas funciones no admiten campos de retorno porque siempre retornan los mismos.
- Separados por espacio o salto de línea
- Si no se indican campos de retorno se dejan sólo las llaves { }

Ejemplos:

```
query{
	getClient(id: "7", plan:"13") {
		id name plan surname createdAt
	}
}

query{
	getClient(
		id: "7"
		plan:"13"
	) {
		id
		name
		plan
		surname
		createdAt
	}
}
```

También toda la consulta se puede escribir en una sola línea:

```
query{ getClient(id:"7", plan:"13") { id name plan surname createdAt } }
```

Puede no solicitarse ninguna respuesta:

```
query{ addClient(id:"7", plan:"13", name:"juanquiroga") {} }
```

Formato de las respuestas de la API


```
{
   "data":{
      "getClient":[{
         "id":"7",
         "name":"Joaquín Sabina",
         "plan":"13",
         "surname":null,
         "createdAt":"2020-04-02"
      }]
   },

   "status":{
      "getClient":{
         "id":"7",
         "done":true,
         "plan":"13",
         "count":1
      }
   }
}
```


- El bloque *data* retorna la información solicitada por la consulta.
- El bloque *status* es información que agrega la API para orientar a la aplicación cliente acerca del procesamiento del resultado, sea que haya sido exitoso o que no haya retornado valores válidos.
  Según la función que se consulte serán los componentes del bloque status y en general son de fácil comprensión, como ser:
  - *id* - confirma el id principal que se usó como parámetro de consulta
  - *count* - reporta el número de registros en la respuesta en funciones que pueden arrojar varios registros
  - *done* - indica si la función fue efectivamente ejecutada
  - *message* - mensajes de utilidad en algunas funciones en particular, por ejemplo *DELETED* o *NONEXISTENT*
- Puede aparecer además un bloque *errors* que brinda más información cuando falla la ejecución de la función


```
 "errors":{
      "getClientDevices":{
         "ID":"NONEXISTENT CLIENT ID"
      }
   }
```

- Tanto *data* como *status* y *errors* contienen sub-bloques encabezados por el nombre de la función o el alias. Si una misma función se consulta varias veces en un mismo query, deben estar precedidas por un *alias* cualquiera.

Ejemplo consulta con *alias*:

```
query{
      g1:getClient(id:"123"){ id name phone }
      g2:getClient(id:"911"){ id firstName lastName }
}
```

Ejemplo respuesta:

```
{
   "data":{
      "g1":[{
         "id":"123",
         "name":"OLGA TAPIAS",
         "phone":""
      }],
      "g2":[{
         "id":"911",
         "firstName":"Thierry",
         "lastName":"Henry"
      }]
   },
   "status":{
      "g1":{
         "id":"123",
         "done":true,
         "count":1
      },
      "g2":{
         "id":"911",
         "done":true,
         "count":1
      }
   }
}
```

FUNCIONES

La API tendrá cada vez más funciones disponibles que serán documentadas aquí explicando en cada función:

1. para qué sirve
2. qué parámetros se le pueden pasar
3. qué campos se le puede pedir que retorne

Pueden ejecutarse 1 o varias funciones en cada query.

Funciones de Configuración

addClient

Agrega un usuario con los datos indicados

```
query{
	addClient(
		id:""
		name:""
		password:""
		firstName:""
		lastName:""
		secondLastName:""
		clientID:""
		phone:""
		phone_2:""
		email:""
		createdAt:""
		plan:""
	){
		id
		name
		password
		firstName
		lastName
		secondLastName
		clientID
		phone
		phone_2
		email
		createdAt
		plan
	}
}
```

- parámetros
  - **id** (ID del usuario)*
  - **name** (login)*
  - **password** (contraseña)
  - **firstName** (nombres)
  - **lastName** (apellido)
  - **secondLastName** (segundo apellido o título - opcional)
  - **clientID** (ID tributaria/personal del cliente opcional)
    Ver información complementaria
  - **phone** (teléfono)
  - **phone_2** (celular)
  - **email** (e-mail)
  - **createdAt** (fecha de alta del usuario)
  - **plan** (grupo al que está asociado el usuario)

- respuesta **data** - campos que se pueden solicitar
  - **id** (ID del usuario - indicado como parámetro, sino asignado automáticamente)
  - **name** (login)
  - **password** (contraseña)
  - **firstName** (nombres)
  - **lastName** (apellido)
  - **secondLastName** (segundo apellido o título - opcional)
  - **clientID** (ID tributaria/personal del cliente opcional)
  - **phone** (teléfono)
  - **phone_2** (celular)
  - **email** (e-mail)
  - **createdAt** (fecha de alta del usuario)
  - **plan** (grupo al que está asociado el usuario)

- respuesta **status**
  - **id** - confirma el ID de usuario agregado (indicado como parámetro, sino asignado automáticamente)
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el resultado de la ejecución de esta función [ADDED|FAILED|MISSING OR WRONG PARAMETERS]

- respuesta **errors**
  retorna la descripción de los errores en la validación de datos enviados. Solo hay respuesta *errors* cuando falla la ejecución del comando.
  Ejemplo de respuesta *errors*:

```
"errors":{
      "addClient":{
         "id":"Client ID is in use by client Ramses V"
      }
   }
```

Ejemplo:

```
query{
	addClient(
		id:"707",
		plan:"13",
		name:"RamsesII"
		password:"123456"
		firstName:"Ramses"
		lastName:"Segundo"
	){
		id
		name
		firstName
		lastName
		plan
		createdAt
	}
}
```

editClient

Modifica los datos de un usuario

```
query{
     editClient(
          id:""
          name:""
          password:""
          firstName:""
          lastName:""
          secondLastName:""
          clientID:""
          phone:""
          phone_2:""
          email:""
          createdAt:""
          plan:""
     ){
          id
          name
          password
          firstName
          lastName
          secondLastName
          clientID
          phone
          phone_2
          email
          createdAt
          plan
     }
}
```

Edita los datos de un usuario según lo indicado en los parámetros y retorna los campos solicitados. Puedes solicitar que retorne sólo los campos modificados, todos, cualesquiera o ninguno.

- parámetros
    - **id** (ID del usuario - si no se especifica el sistema asignará un ID automáticamente)
    - **name** (login)*
    - **password** (contraseña)*
    - **plan** (grupo al que está asociado el usuario)
    - **firstName** (nombres)

  - **lastName** (apellido)
  - **secondLastName** (segundo apellido o título - opcional)
  - **clientID** (ID tributaria/civil del cliente opcional)
    Ver información complementaria
  - **phone** (teléfono)
  - **phone_2** (celular)
  - **email** (e-mail)
  - **createdAt** (fecha de alta del usuario)

- respuesta **data** - campos que se pueden solicitar
  - **id** (ID del usuario)
  - **name** (login)
  - **plan** (grupo al que está asociado el usuario)
  - **password** (contraseña)
  - **firstName** (nombres)
  - **lastName** (apellido)
  - **secondLastName** (segundo apellido o título - opcional)
  - **clientID** (ID tributaria/civil del cliente opcional)
  - **phone** (teléfono)
  - **phone_2** (celular)
  - **email** (e-mail)
  - **createdAt** (fecha de alta del usuario)

- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el resultado de la ejecución de esta función
    [UPDATED|UNMODIFIED|MISSING OR WRONG PARAMETERS|FAILED]

- respuesta **errors**
  retorna la descripción de los errores en la validación de datos enviados. Solo hay respuesta *errors* cuando falla la ejecución del comando.
  Ejemplo de respuesta *errors*:

```
"errors":{
      "editClient":{
         "name":"Client login is in use by client 758",
         "plan":"Group ID not valid"
      }
}
```

Ejemplo:

```
query{
      editClient  (
            id:"33",
            name: "Juanica",
            password: "123456",
            plan:"11",
            firstName:"Juana",
            lastName:"de Arco",
            secondLastName:"",
            clientID:"3519044172",
      ) {
            id
            name
            plan
            firstName
            lastName
            clientID
            password
      }
}
```

delClient

Elimina un usuario y todas sus conexiones

```
query{
      delClient( id:"2021" ){ }
}
```

- parámetros
  - **id** - ID (número) de usuario*
- respuesta **data**
  - sólo retorna el resultado de la ejecución de la función [true|false]
- respuesta **status**
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el estado de los datos luego de ejecutar la función [DELETED|NONEXISTENT||FAILED]
- respuesta **errors**
  Repite el mismo valor de *message* sólo cuando la ejecución falla.
  Ejemplo de respuesta *error*:

```
"errors":{
      "delClient":{
         "id":"NONEXISTENT"
      }
}
```

Ejemplos:

```
query{
      delClient(id:"299"){}
}

query{
      c1:delClient(id:"299"){}
      c1:delClient(id:"54"){}
}
```

getClient

Obtiene los datos de uno o varios usuarios

```
query{
     getClient(
          id:""
          plan:""
          status:""
          clientID:""
     ){
          id
          name
          password
          firstName
          lastName
          secondLastName
          clientID
          phone
          phone_2
          email
          createdAt
          plan
     }
}
```

- parámetros (todos son opcionales)
  - **id** - ID (número) de usuario. Si se deja vacío, retorna todos los usuarios.
  - **plan** - ID del Grupo al que está asociado el/los usuario/s. Si se deja vacío retorna usuarios de cualquier Grupo.
  - **status** - Si es *true* sólo retorna usuarios habilitados y con conexiones activas.
  - **clientID** - Puedes consultar por el número de identificación personal/tributaria - Ver información complementaria
- respuesta **data** - campos que se pueden solicitar
  - **id** (ID del usuario)
  - **name** (login)
  - **password** (contraseña)
  - **firstName** (nombres)
  - **lastName** (apellido)
  - **secondLastName** (segundo apellido o título - opcional)
  - **clientID** (ID tributaria/personal del cliente opcional)
  - **phone** (teléfono)
  - **phone_2** (celular)

    - **email** (e-mail)
    - **createdAt** (fecha de alta del usuario)
    - **plan** (grupo al que está asociado el usuario)

- respuesta **status**
    - **id** | **plan** | **status** - confirma los parámetros utilizados en la consulta
    - **done** - confirma si se realizó el query [true|false]
    - **count** - cantidad de registros (usuarios) en la respuesta

Ejemplo:

```
query{

	c1:getClient(){
		id
		name
		plan
		createdAt
	}

	c2:getClient(id: "7"){
		id
		name
		firstName
		lastName
		email
		plan
	}

	c3:getClient(plan: "3", status:true){
		id
		name
		firstName
		lastName
		email
	}
}
```

addClientDevice

Agrega una conexión a un usuario

```
query{
      addClientDevice(
            id:""
            status:true|false
            byLogin:""
            byLoginUnique:""
            ip:""
            mac:""
            publicIP:""
            byDate:true|false
            start:""
            end:""
            startAdvanced:""
            renew:""
            byLapse:true|false
            lapseDays:""
            lapseHours:""
            lapseMinutes:""
            noDhcp:true|false
      ){
            id
            connection
            byLogin
            byLoginUnique
            ip
            mac
            publicIP
            lastModified
            byDate
            start
            end
            startAdvanced
            renew
            status
            byLapse
            lapseDays
            lapseHours
            lapseMinutes
            noDhcp
            logged
      }
}
```

- parámetros
    - Identificación
        - **id** - ID del usuario [256]*
    - Modalidad de autenticación
        - **byLogin** - indica si la conexión se activa con el login [true|false]
        - **byLoginUnique** -se fija la conexión al primer equipo desde el cual se ha logueado - no puede loguearse nuevamente desde otro equipo [true|false]
        - **ip** - IP de la conexión [192.168.2.35]
        - **mac** - MAC de la conexión [00:11:AB:25:F4:D5]
    - Validez de la conexión
        - **byDate** - indica si aplica una fecha límite de inicio y de finalización para la conexión [true|false]
            - **start** - fecha de inicio del periodo habilitado de conexión [2023-12-30]
            - **end** - fecha de finalización del periodo habilitado de conexión [2023-12-30]
            - **startAdvanced** - en el caso que la conexión tenga una fecha **start** en el futuro, aquí se puede indicar que la primera vez se inicie en una fecha anterior [2023-12-30]
        - **byLapse** - indica si aplica un lapso de tiempo en que la conexión estará vigente a partir del login [true|false]
            - **lapseDays** - días del lapso de vigencia [10]
            - **lapseHours** - horas del lapso de vigencia [12]
            - **lapseMinutes** - minutos del lapso de vigencia [45]
    - Estado y renovación
        - **status** - estado activo o pausado de la conexión [true|false]
        - **publicIP** - IP pública direccionada a la ip privada de la conexión [200.45.79.23]
        - **renew** - criterio que utilizará el sistema cuando se le solicite la renovación de una conexión con fecha vencida (o sea, **byDate** *true* y fecha **end** ya pasada) - Asignando el valor *lapse_day* las conexiones se renuevan tomando en cuenta la cantidad de días entre end y start contados a partir del momento que se renueva; asignando *month_day* la conexión se renueva tomando el mismo día del mes que tiene start contando en el mes en curso o el mes siguiente si ese día ya pasó [lapse_day|month_day]
        - **noDhcp** - evita que esta IP con su MAC sean incluidas en la configuración del servicio DHCP de Sislander (opción poco utilizada ya que no perjudica que se incluya en el DHCP por más que el cliente tenga configuración fija y no lo utilice) [true|false]
    - Observaciones, excepciones y compatibilidades

- Si solo se especifica el **id** del usuario, se creará una conexión por login sin ninguna otra característica
- Si **byLogin** es *true* se genera una conexión por login, anulando los valores de **ip**, **mac**, **dhcp**
- **byLapse** sólo tiene sentido y validez si **byLogin** es *true*
- Si **byLapse** no es *true*, **lapseDays**, **lapseHours** y **lapseMinutes** son ignorados
- Se pueden especificar uno solo, dos cualesquiera o los tres valores de **lapseDays**, **lapseHours** y **lapseMinutes** que sumados conformarán el tiempo de validez de la conexión a partir del login
- **publicIP** puede asignarse a conexiones por ip y mac y también a conexiones por login
- Si **byDate** no es *true*, **start** y **end** son ignorados
- Pueden combinarse **byDate** y **byLapse** en conexiones por login. Ejemplo: se permite navegar por 7 días a partir del login dentro del mes de abril

```
query{
  addClientDevice(
    id:"1"
    byLogin:true
    byLapse:true
    lapseDays:"7"
    byDate:true
    start:"2021-04-01"
    end:"2021-04-30"
  ){
    id connection byLapse lapseDays byDate start end
  }
}
```

- respuesta **data** - campos que se pueden solicitar (los mismos campos utilizados como parámetros y 3 más que solo están disponibles en la respuesta)
  - **id** - ID del usuario
  - **connection** - ID automático de la conexión agregada
  - **byLogin** - si el cliente debe activar la conexión con su login
  - **byLoginUnique** - si la conexión activada por login quedará fija al primer equipo y no puede activarlo en otro más adelante
  - **logged** - si una conexión ya fue activada por login
  - **ip** - IP de la conexión
  - **mac** - MAC de la conexión
  - **noDhcp** - si evita que el DHCP configure esta IP y MAC

- **publicIP** - IP pública de la conexión
- **byDate** - si aplica las fechas start y end
- **start** - fecha de inicio del periodo habilitado de conexión
- **end** - fecha de finalización del periodo habilitado de conexión
- **byLapse** - si aplica un lapso de tiempo de vigencia a partir del login
- **lapseDays** - días del lapso de vigencia
- **lapseHours** - horas del lapso de vigencia
- **lapseMinutes** - minutos del lapso de vigencia
- **renew** - criterio de renovación para conexiones con **start** y **end** ya vencidas
- **status** - estado activo o pausado
- **lastModified** - fecha de modificación
- **status** - estado de la conexión
  Son los mismos campos de retorno que retorna la función setClientDevicesStatus (consultar el detalle de esa función para más información). Aquí retornarán en un subarray, como en el siguiente ejemplo (fragmento de respuesta):

```
"status":{
      "status_enabled":true,
      "status_active":false,
      "status_unactive_reason":"GROUP DISABLED"
}
```

- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el resultado de la ejecución de esta función [ADDED|FAILED|MISSING OR WRONG PARAMETERS]
  - **description** - informa errores en los parámetros que han impedido agregar el usuario

- respuesta **errors**
  retorna la descripción de los errores en la validación de datos enviados. Solo hay respuesta *errors* cuando falla la ejecución del comando.

Ejemplo de respuesta *errors*:

```
"errors":{
      "addClientDevice":{
         "1":" Usuarios con esta IP asignada permanentemente 1 JuanLopez",
         "2":"IP en uso.",
         "3":"Esta IP publica no puede aplicarse sobre ninguna de las redes
externas configuradas. Red 1 Internet WAN1 con IP 10.45.45.11 y máscara de red
255.255.255.0 permite IPs entre la 10.45.45.1 y la 10.45.45.254 excepto la
misma 10.45.45.11 y la 10.45.45.1 que es la puerta de enlace."
      }
}
```

Ejemplos:

```
query{
	addClientDevice(
		id:"1"
		ip:"10.44.44.25"
		mac:"00:11:22:33:44:55"
		publicIP:"200.45.79.23"
		byDate:true
		start:"2021-09-01"
		end:"2021-09-30"
		startAdvanced:"2021-08-20"
	){
		id
		connection
		byLogin
		byLoginUnique
		ip
		mac
		publicIP
		lastModified
		byDate
		start
		end
		startAdvanced
		renew
		status
		byLapse
		lapseDays
		lapseHours
		lapseMinutes
		noDhcp
		logged
	}
}
```

editClientDevice

Edita una conexión de un usuario

```
query{
      editClientDevice(
            id:""
            connection:""
            status:true|false
            byLogin:""
            byLoginUnique:""
            ip:""
            mac:""
            publicIP:""
            byDate:true|false
            start:""
            end:""
            startAdvanced:""
            renew:""
            byLapse:true|false
            lapseDays:""
            lapseHours:""
            lapseMinutes:""
            noDhcp:true|false
      ){
            id
            connection
            byLogin
            byLoginUnique
            ip
            mac
            publicIP
            lastModified
            byDate
            start
            end
            startAdvanced
            renew
            status
            byLapse
            lapseDays
            lapseHours
            lapseMinutes
            noDhcp
            logged
      }
}
```

- **parámetros**
  Consultar addClientDevice ya que son los mismos parámetros con las mismas funciones.
  editClientDevice admite y en ciertos casos requiere un parámetro más:
  - **connection** - ID de la conexión que se quiere editar*
    Cuando se agrega una conexión de usuario el sistema le asigna automáticamente un ID de conexión que se puede consultar en la misma respuesta de addClientDevice así como con getClientDevices.
    Cuando quieres modificar los datos de una conexión:
    - Si el cliente tiene una sola conexión puede no indicarse **connection**; el sistema automáticamente modificará su única conexión con los parámetros que envías.
    - Si el cliente tiene varias conexiones se debe indicar **connection** (el ID de conexión que estás modificando) con editClientDevice.
    - Si el cliente tiene varias conexiones y no se indica **connection** en los parámetros de editClientDevice no se hará ninguna modificación, no se ejecutará la función.

- respuesta **data** - campos que se pueden solicitar
  Consultar addClientDevice ya que son los mismos campos de retorno.

- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el resultado de la ejecución de esta función
  - **status** - estado de la ejecución de la función [UPDATED]

- ***errors*** informa los errores encontrados cuando la función no puede ser ejecutada

Ejemplos:

```
query{
	editClientDevice(
		id:"1"
		connection:"763"
		byLogin:true
		byLapse:true
		lapseDays:"7"
		byDate:true
		start:"2021-04-01"
		end:"2021-04-30"
	){
		id
		connection
		byLogin
		byLoginUnique
		byDate
		start
		end
		startAdvanced
		byDateRenew
		status
		byLapse
		lapseDays
		lapseHours
		lapseMinutes
	}
}
```

delClientDevice

Elimina una conexión de un usuario

```
query{
     delClientDevice(
          id:""
          connection:""
     ){
     }
}
```

- **parámetros**
  - **id** - ID del cliente (usuario) dueño de la conexión que se quiere eliminar *
  - **connection** - ID de la conexión que se quiere eliminar *

- respuesta **data**
  No hay campos para solicitar. En forma predeterminada retorna una respuesta indicando el ID de la conexión eliminada y el ID del usuario al que pertenecía la conexión.

```
"data":{
      "delClientDevice":{
         "id":"911",
         "connection":"772"
      }
   },
```

- respuesta **status**
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el resultado de la ejecución de esta función

```
"status":{
   "delClientDevice":{
      "done":true,
      "message":"DELETED"
   }
}
```

- ***errors*** informa los errores encontrados cuando la función no puede ser ejecutada repitiendo el valor de message. La aplicación que utiliza la API puede detectar errores simplemente por las respuestas de *status* o analizando *errors* como en otras funciones.

```
"errors":{
    "delClientDevice":["INVALID CLIENT ID"]
}
```

Ejemplo:

```
query{
      delClientDevice(id:"911", connection:"772"){}
}
```

getClientDevices

Obtiene los datos de la conexión indicada o de todas las conexiones de un usuario

```
query{
     getClientDevices(
          id:""
          connection:""
     ){
          id
          connection
          byLogin
          byLoginUnique
          ip
          mac
          publicIP
          lastModified
          byDate
          start
          end
          startAdvanced
          renew
          status
          byLapse
          lapseDays
          lapseHours
          lapseMinutes
          noDhcp
          logged
          lapseEnd
     }
}
```

- parámetros
  - **id** - ID (número) de usuario*
  - **connection** - ID de conexión
    - si no se indica *connection*, retorna todas las conexiones del cliente
    - si se indica *connection* y esa ID de conexión no existe o no está asociado al cliente lo informa en *message*

- respuesta **data** - campos que se pueden solicitar
  Esta función puede retornar los mismos campos que retorna la función addClientDevice. Consultar dicha función por más detalles sobre cada parámetro. Además, agrega 2 campos que no se utilizan al agregar o editar una conexión (ver al final de la lista).

    - id
    - connection
    - byLogin
    - byLoginUnique
    - ip
    - mac
    - publicIP
    - lastModified
    - byDate
    - start
    - end
    - startAdvanced
    - renew
    - status
    - byLapse
    - lapseDays
    - lapseHours
    - lapseMinutes
    - noDhcp
    - status
    - **logged**
      En las conexiones byLogin indica si el usuario ya se ha logueado y activado la conexión
    - **lapseEnd**
      En las conexiones byLapse ya logueadas, indica el día/hora en que finaliza o finalizó el periodo de navegación.
      La conexión se bloquea en el día/hora *lapseEnd* de una conexión *byLapse=true* o en el día/hora *end* de una conexión *byDate=true*. Lo que suceda primero si la conexión tiene ambos parámetros.

- respuesta **status**
    - **id** - confirma el ID de usuario consultado
    - **count** - cantidad de registros (usuarios) en la respuesta
    - **done** - confirma si se realizó la consulta [true|false]
    - **message** - mensajes adicionales en caso de respuestas excepcionales [NO CONNECTION ID FOR THIS CLIENT ID|NO CONNECTIONS FOR THIS CLIENT ID|CLIENT ID ERROR]

- ***errors*** informa los errores encontrados cuando la función no puede ser ejecutada -
    - *errors* aparece principalmente si el *id* del cliente es inválido o el cliente no existe en el sistema
    - no informa *errors* si el *id* cliente es válido pero el mismo no tiene conexiones (no es un error sino un resultado vacío)

Ejemplos:

```
query{
      getClientDevices(id:"4"){
            id
            connection
            byLogin
            byLoginUnique
            ip
            mac
            publicIP
            createdAt
            byDate
            start
            end
            startAdvanced
            byDateRenew
            status
            byLapse
            lapseDays
            lapseHours
            lapseMinutes
            noDhcp
            logged
            lapseEnd
      }
}
```

renewClientDevices

Renueva las fechas de inicio y fin de de la conexión indicada o de todas las conexiones de un usuario

```
query{
     renewClientDevices(
          id:""
          connection:""
     ){
          id
          connection
          byLogin
          byLoginUnique
          ip
          mac
          publicIP
          lastModified
          byDate
          start
          end
          startAdvanced
          renew
          status
          byLapse
          lapseDays
          lapseHours
          lapseMinutes
          noDhcp
          logged
          lapseEnd
     }
}
```

Renueva el periodo activo de las conexiones de un usuario.
Por otra parte, si la conexión no está habilitada, la habilita, lo cual equivale a ejecutar *setClientDevicesStatus* con *status:true*.

-

- parámetros
    - **id** - ID (número) de usuario*
    - **connection** - ID de conexión
        - si no se indica *connection*, renueva todas las conexiones del cliente
        - si *connection* es pasada como parámetro, renueva solamente esa conexión del cliente y retorna los datos de esa conexión
        - si se indica *connection* y esa ID de conexión no existe o no está

asociado al cliente lo informa en *errors*

- respuesta **data** - campos que se pueden solicitar
  Esta función puede retornar los mismos campos que retorna la función addClientDevice. Consultar dicha función por más detalles sobre cada parámetro. Además, agrega 2 campos que no se utilizan al agregar o editar una conexión (ver al final de la lista).
  - id
  - connection
  - byLogin
  - byLoginUnique
  - ip
  - mac
  - publicIP
  - lastModified
  - byDate
  - start
  - end
  - startAdvanced
  - renew
  - status
  - byLapse
  - lapseDays
  - lapseHours
  - lapseMinutes
  - noDhcp
  - **logged**
    En las conexiones byLogin indica si el usuario ya se ha logueado y activado la conexión
  - **lapseEnd**
    En las conexiones byLapse ya logueadas, indica el día/hora en que finaliza o finalizó el periodo de navegación.
    Una conexión se bloqueará en el día/hora *lapseEnd* de una conexión con *byLapse=true* o en el día/hora *end* de una conexión con *byDate=true*. Lo que suceda primero si la conexión tiene ambos parámetros.

- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **count** - cantidad de registros (usuarios) en la respuesta
  - **done** - confirma si se realizó la consulta [true|false]

- ***errors*** informa los errores encontrados cuando la función no puede ser ejecutada -
  - *errors* aparece principalmente si el *id* del cliente es inválido o el cliente no

existe en el sistema
- no informa *errors* si el *id* cliente es válido pero el mismo no tiene conexiones (no es un error sino un resultado vacío)

Ejemplos:

```
query{
	renewClientDevices(id:"4"){
		id
		connection
		byLogin
		byLoginUnique
		ip
		mac
		publicIP
		createdAt
		byDate
		start
		end
		startAdvanced
		byDateRenew
		status
		byLapse
		lapseDays
		lapseHours
		lapseMinutes
		noDhcp
		status
	}
}
```

setClientDevicesStatus

Modifica el estado habilitado o deshabilitado de las conexiones de un usuario

```
query {
     setClientDevicesStatus(id:"4", status:true){}
}
```

- parámetros
  - **id** - ID (número) de usuario*
  - **status** - estado habilitado de una conexión o de todas las conexiones del usuario [true|false]*
  - **connection** - ID de conexión
    - si no se indica *connection*, establece el estado de todas las conexiones del cliente
    - si se indica connection, establece el estado solo de la conexión del cliente indicada
    - si se indica *connection* y esa ID de conexión no existe o no está asociado al cliente lo informa emite *errors*

- respuesta **data**
  retorna una respuesta por cada conexión encontrada y procesada, cada una identificada por su ID de conexión

  - **status_enabled**
    Indica si la conexión está habilitada
    Valores posibles:
    - true: habilitado
    - false: no-habilitado
    - null: usuario o conexión no encontrados

  - **status_active**
    Indica si la conexión está activa [true|false]
    Una conexión puede estar habilitada para funcionar (status: true) pero aún así no estar activa por varios motivos, como ser: a) tiene una fecha de inicio en el futuro, b) es conexión por login y aún no se ha logueado el cliente, c) tiene un lapso de conexión a partir del login que ya expiró y otras. Una conexión no habilitada (status: false) siempre estará inactiva.

  - **status_unactive_reason**
    Explica porqué la conexión está inactiva
    Valores posibles:
    - EXPIRED: la conexión tiene fecha de finalización ya vencida
    - FUTURE: la conexión tiene fecha de inicio aún no iniciada (en el futuro)
    - UNLOGGED: es una conexión por login y no se ha autenticado el cliente

- USER DISABLED: las conexiones pertenecen a un usuario no habilitado
- GROUP DISABLED: las conexiones pertenecen a un usuario asignado a un grupo no habilitado
- NETWORK DISABLED: las conexiones pertenecen a un usuario asignado a un grupo asociado a una Red LAN no habilitada
- CONNECTION DISABLED: siempre se usa este valor en *status_unactive_reason* cuando la conexión no está habilitada (*status_enabled false*) y por lo tanto tampoco está activa (*status_active* false)

Una conexión puede estar inactiva por varios motivos a la vez (por ejemplo conexión por login aún no logueada, con fecha de inicio en el futuro, perteneciente a un usuario no habilitado) en cuyo caso se reportará todas las causas separadas por | .

- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **done** - confirma si se realizó la consulta [true|false]
  - **count** - cantidad de conexiones del usuario encontradas y procesadas

- respuesta **errors**
  retorna la descripción de los errores en la validación de datos enviados. Solo hay respuesta *errors* cuando falla la ejecución del comando porque el usuario no existe, el ID de usuario no es válido o el ID de conexión enviado en los parámetros no es válido o dicha conexión no pertenece a este usuario
  Ejemplo de respuesta *errors*:

```
"errors":{
   "getClientStatus":{
      "id":"INVALID CLIENT ID"
   }
}
```

Ejemplos:

```
query{
      s1:setClientDevicesStatus(id: "44", status:true ){ }
      s2:setClientDevicesStatus(id: "1", connection:"761", status:false ){ }
}
```

setClientStatus

Establece el estado habilitado o deshabilitado de todas las conexiones de un usuario a la vez

```
query {
     setClientStatus(id:"4", status:true){}
}
```

Es una alias de setClientDevicesStatus con los mismos parámetros y respuestas. Generalmente se utiliza para habilitar/deshabilitar todas las conexiones del usuario.

getClientDevicesStatus

Obtiene el estado habilitado o deshabilitado de una o de todas las conexiones de un usuario

```
query {
      getClientDevicesStatus(id:"4", connection:"816"){}
}
```

Una conexión puede estar habilitada para funcionar o deshabilitada (pausada).
Una conexión habilitada puede no estar activa por varios motivos como se explica en la sección en setClientDevicesStatus.
Esta función retorna ambos estados: el habilitado|deshabilitado y el activo|inactivo.

- parámetros
  - **id** - ID (número) de usuario*
  - **connection** - ID de conexión
    - si no se indica *connection*, consulta el estado de todas las conexiones del cliente
    - si se indica connection, consulta el estado solo de la conexión del cliente indicada
    - si se indica *connection* y esa ID de conexión no existe o no está asociado al cliente emite *errors*

- respuesta **data**
  Consultar setClientDevicesStatus ya que son los mismos campos de respuesta.

- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **done** - confirma si se realizó la consulta [true|false]
  - **count** - cantidad de conexiones del usuario encontradas

- respuesta **errors**
  Consultar setClientDevicesStatus ya que son los mismos campos de error.

Ejemplos:

```
query{
      s1:getClientDevicesStatus(id: "44"){ }
      s2:getClientDevicesStatus(id: "1", connection:"761"){ }
}
```

getClientStatus

retorna el estado de conexión de un usuario.

```
query {
     getClientStatus(id:"4", status:true){}
}
```

Consulta todas las conexiones de un usuario y
➔ si encuentra al menos una conexión habilitada retorna *status_enabled true*
➔ si encuentra al menos una conexión activa retorna *status_active true*

- parámetros
  - **id** - ID (número) de usuario*
- respuesta **data**
  - **status_enabled**
    Indica si el usuario tiene al menos una conexión habilitada
  - **status_active**
    Indica si el usuario tiene al menos una conexión activa

  Consultar información sobre estos campos de la respuesta data en setClientDevicesStatus ya que funcionan de igual forma.
- respuesta **status**
  - **id** - confirma el ID de usuario consultado
  - **done** - confirma si se realizó la consulta [true|false]
  - **count** - cantidad de conexiones del usuario encontradas
- respuesta **errors**
  Consultar setClientDevicesStatus ya que son los mismos campos de error.

  Retorna los mismos campos que getClientDevicesStatus pero con los mismos parámetros y respuestas.

getPlan

Obtiene los datos de uno o varios grupos de Sislander (los grupos equivalen a los planes de servicio)

```
query{
     getPlan(
          id:""
     ){
          id
          name
          wanId
          lanId
          lanNic
          bwDown
          bwBurst
          bwUp
          bwMax
          bwLocal
          balancedHttp
          active
     }
}
```

- parámetros (todos son opcionales)
  - **id** - ID (número) del grupo/plan. Si se deja vacío retorna todos los grupos/planes.
- respuesta **data** - campos que se pueden solicitar
  - **id** - ID del grupo/plan
  - **name** - nombre con el cual se conoce el plan
  - **wanId**
    ID de red WAN por el cual se conectan todos los usuarios que tienen este plan. Si está vacía se conectan a todas las WAN principales, o sea, utilizan balanceo de conexiones a internet)
  - **lanNic**
    Tarjeta/dispositivo de red por el cual conectan al servidor los usuarios de este plan. Por el momento, cada plan solo puede asociarse a una tarjeta de red por lo cual hay que configurar distintos planes para cada tarjeta LAN. Debe haber por lo menos una Red Lan configurada en el sistema que utilice esta tarjeta de red (lanNic).

- **lanId**
  Id de una red LAN cualquiera que utiliza la tarjeta de red lanNic. Al Plan lo que le importa es por cual tarjeta se conectan los usuarios. La lanId está asociada a alguna tarjeta de red. Por lo tanto al conocer la lanId podemos conocer la tarjeta de red por la cual deben conectarse los usuarios de este plan.
- **bwDown**
  Máximo ancho de banda de bajada autorizado a cada usuario de este plan.
- **bwBurst**
  Ancho de bajada adicional en forma de burst (efecto flash) para acceso rápido a las páginas web. Los valores posibles son: 1 Sin efecto flash, 2 Flash básico, 3 Flash estándar, 4 Flash máximo, 5 Sitio web medio, 6 Sitio web full, 7 Automático - siendo 7 el valor por defecto. Las explicaciones sobre cada valor se encuentran en el mismo sistema de control de Sislander en el formulario de Edición de Grupos.
- **bwUp**
  Máximo ancho de banda de subida autorizado a cada usuario de este plan.
- **bwMax**
  Máximo ancho de banda de bajada autorizado para todos los usuarios del plan en su conjunto. Permite en la práctica establecer un reuso máximo entre los usuarios del plan. Habitualmente se deja vacío porque la distribución proporcional de Sislander administra automáticamente el reuso, sin necesidad de limitar el consumo máximo en conjunto de los usuarios por grupo.
- **bwLocal**
  Máximo ancho de banda de bajada autorizado a cada usuario de este plan para las transferencias locales, del caché y otras que estén marcadas en esta categoría. Habitualmente se deja vacío porque es ancho de banda libre.
- **balancedHttp**
  Cuando los usuarios del plan se conectan por alguna wanID en particular y no balancean, se puede indicar aquí que la navegación http sí sea balanceada para que así pase por el proxy, el cual siempre utiliza el balanceo. Este parámetro no tiene efecto cuando la wanID está vacía o sea cuando los usuarios del grupo no salen por una WAN en particular sino que entran en el balanceo de conexiones.
- **active** - indica si el plan y por lo tanto también sus usuarios están activos

- respuesta **status**
  - **id** - confirma el parámetro id utilizados en la consulta
  - **done** - confirma si se realizó el query [true|false]
  - **count** - cantidad de registros (planes/grupos) en la respuesta

Ejemplo:

```
query{
	p1:getPlan(
		id:"10"
	){
		id
		name
		wanId
		lanId
		lanNic
		bwDown
		bwBurst
		bwUp
		bwMax
		bwLocal
		balancedHttp
		active
	}

	p2:getPlan(
		id:"9"
	){
		id
		name
		wanId
		lanId
		lanNic
		bwDown
		bwUp
		active
	}

	p3:getPlan()
	{
		id
		name
		bwDown
		bwUp
		active
	}
}
```

editPlan

Modifica los datos de configuración de un grupo/plan

```
query{
     editPlan(
          id:""
          name:""
          wanId:""
          lanId:""
          lanNic:""
          bwDown:""
          bwBurst:""
          bwUp:""
          bwMax:""
          bwLocal:""
          balancedHttp:""
          active:""
     ){
          id
          name
          wanId
          lanId
          lanNic
          bwDown
          bwBurst
          bwUp
          bwMax
          bwLocal
          balancedHttp
          active
     }
}
```

- parámetros (todos son opcionales)
    - **id** - ID (número) del grupo/plan es obligatorio. Debe identificarse cuál plan se va a editar. El plan debe existir.
    - Todos los demás parámetros son opcionales, según lo que se necesite modificar.
    - Consultar getPlan ya que son los mismos parámetros con las mismas funciones.
    - Observaciones especiales para edición sobre los parámetros *lanId* y *lanNic*:

        - *lanId* y *lanNic* son en cierto modo equivalentes. Le indican al Grupo/Plan por cual tarjeta de red (Nic) se conectan sus usuarios.
        - Si indicas *lanID* esta debe ser una Red LAN válida ya configurada en el sistema y el Plan se aplicará sobre la tarjeta de red (nic) configurada para esta Red
        - Si, en cambio, indicas *lanNic*, debe existir previamente configurada en el sistema alguna Red LAN cualquiera que utilice esta tarjeta de red (nic) y entonces el Plan quedará asociado a cualquier Red LAN que utilice esta tarjeta
        - Es lo mismo indicar *lanId* o *lanNic* mientras quede establecida la tarjeta de red que corresponda a los usuarios de este Grupo/Plan.
    - Los cambios en los Grupos/Planes y por lo tanto en sus usuarios, solamente tienen efecto cuando reinicias el control general. Desde la API se puede hacer con la función *restartControl*

- respuesta **data** - campos que se pueden editar
    - Consultar getPlan ya que son los mismos parámetros con las mismas funciones
    - **id** es la excepción ya que una vez creado el Grupo/Plan, este no se puede modificar

- respuesta **status**
    - **id** - confirma el parámetro id utilizados en la consulta
    - **done** - confirma si se realizó el query [true|false]
    - **message** - informa el resultado de la ejecución de esta función

- respuesta **errors**
  retorna la descripción de los errores en la validación de datos enviados. Solo hay respuesta *errors* cuando falla la ejecución del comando.
  Ejemplo de respuesta *errors*:

```
"errors":{
    "editPlan":{
        "name":"Group name required",
        "lanId":"incorrect lan id",
        "bwDown":"Incorrect download bandwidth",
        "bwUp":"incorrect upload bandwidth"
    }
}
```

Ejemplo:

```
query{
	editPlan(
		id:"9"
		name:"MaxSpeed 35M"
		wanId:"1"
		lanNic:"eth3"
		bwDown:"35000"
		bwUp:"15000"
		bwBurst:"3"
		bwMax:""
		bwLocal:""
		balancedHttp:true
		active:true
	){
		id
		name
		wanId
		lanId
		lanNic
		bwDown
		bwBurst
		bwUp
		bwMax
		bwLocal
		balancedHttp
		active
	}
}
```

addPlan

Agrega un grupo/plan

```
query{
     addPlan(
          id:""
          name:""
          wanId:""
          lanId:""
          lanNic:""
          bwDown:""
          bwBurst:""
          bwUp:""
          bwMax:""
          bwLocal:""
          balancedHttp:""
          active:""
     ){
          id
          name
          wanId
          lanId
          lanNic
          bwDown
          bwBurst
          bwUp
          bwMax
          bwLocal
          balancedHttp
          active
     }
}
```

- parámetros (todos son opcionales)
    - **id** - ID (número) del grupo/plan es opcional. Si no indicas un ID, el sistema asignará el siguiente ID al grupo con ID más alto. Si indicas un ID, no debe pertenecer a ningún grupo existente.
    - Puedes indicar ya sea *lanId* o *lanNic* con las mismas consideraciones que se explican para editPlan
    - Datos obligatorios para agregar un grupo/plan:
        - name

        - lanId o lanNic
        - bwDown
        - bwUp
        - active (no es obligatorio pero si no lo indicas, el grupo no queda habilitado, si bien luego puedes modificarlo con editPlan)
    - Si no se indica *wanId*, se asocia el grupo con todas las Redes WAN que participen del balanceo (permanentes)
    - Consultar editPlan ya que son los mismos parámetros con las mismas funciones.
    - Los nuevos Grupos/Planes y los cambios en los usuarios a los que se los asigna, solamente tienen efecto cuando reinicias el control general. Desde la API se puede hacer con la función *restartControl*
- respuesta **data** - campos que se pueden editar
    - Consultar getPlan ya que son los mismos parámetros con las mismas funciones

- respuesta **status**
    - **id** - confirma el parámetro id utilizados en la consulta
    - **done** - confirma si se realizó el query [true|false]
    - **message** - informa el resultado de la ejecución de esta función

- respuesta **errors**
  retorna la descripción de los errores en la validación de datos enviados. Solo hay respuesta *errors* cuando falla la ejecución del comando.
  Ejemplo de respuesta *errors*:

```
"errors":{
    "editPlan":{
        "name":"Group name required",
        "lanId":"incorrect lan id",
        "bwDown":"Incorrect download bandwidth",
        "bwUp":"incorrect upload bandwidth"
     }
}
```

Ejemplo:

```
query{
	addPlan(
		id:"9"
		name:"MaxSpeed 35M"
		lanNic:"eth3"
		bwDown:"35000"
		bwUp:"15000"
		active:true
	){
		id
		name
		wanId
		lanId
		lanNic
		bwDown
		bwBurst
		bwUp
		bwMax
		bwLocal
		balancedHttp
		active
	}
}
```

delPlan

Elimina un grupo/plan

```
query{
      delPlan( id:"" ){ }
}
```

- parámetros
  - **id** - ID (número) del grupo/plan*
- respuesta **data**
  - sólo retorna el resultado de la ejecución de la función [true|false]
- respuesta **status**
  - **done** - confirma si se realizó el query [true|false]
  - **message** - informa el estado de los datos luego de ejecutar la función [DELETED|NONEXISTENT|FAILED|USERS ASSIGNED|ID REQUIRED]
    - DELETED (el grupo/plan fue eliminado)
    - NONEXISTENT (el grupo/plan no existe en la configuración)
    - USERS ASSIGNED X (no se puede eliminar porque hay X cantidad de usuarios asociados a este grupo/plan)
    - ID REQUIRED (falta especificar el Id de plan en los parámetros)
    - FAILED (se ejecutó el proceso pero el grupo sigue existiendo - fallo)
- respuesta **errors**
  Repite el mismo valor de *message* sólo cuando la ejecución falla.
  Ejemplo de respuesta *error*:

```
"errors":{
      "delPlan":{
         "id":"NONEXISTENT"
      }
}
```

Ejemplos:

```
query{
      delPlan(id:"299"){}
}

query{
      c1:delPlan(id:"299"){}
      c1:delPlan(id:"54"){}
}
```

Funciones de Control

restartControl

Reinicia el control general

```
query{
     restartControl(){ }
}
```

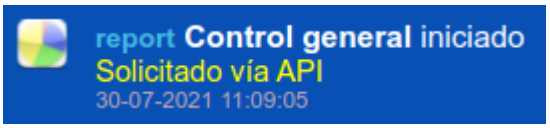
Reinicia si está activo o *inicia* si está detenido el Control General, o sea, el control de acceso y el control de ancho de banda. En este proceso también siempre se reinician las configuraciones de red y -si están activos- el firewall y la redirección de IPs públicas.

- parámetros
  - no requiere ningún parámetro
- respuesta **data**
  - *true* si se ejecutó el proceso y el control general quedó en estado activo
  - *false* si no fue posible ejecutar el reinicio (por ejemplo si el administrador del sistema u otra llamada de API ya estaba reiniciando el control general) o si a pesar de haberse ejecutado, por alguna condición del sistema o de la comunicación, el control general no quedó en estado activo
- respuesta **status**
  - **done** - confirma si se ejecutó el reinicio, independientemente del resultado (si quedó activo o no) [true|false]
  - **message** - informa qué sucedió al ejecutar la función de control
    - STARTED: ejecutó la función de control y reinició el control general
    - TRIED: ejecutó la función de control pero no pudo reiniciar el control general
    - FAILED: no pudo ejecutar la función de control
  - **control** - informa el estado en el que quedó el control general, más allá de si se ejecutó o no la función de control [ACTIVE|UNACTIVE]
- respuesta **errors**
  Repite el mismo valor de *message* dentro de la variable *process,* sólo cuando la ejecución falla.

Ejemplo de respuesta *error*:

```
"errors":{
      "restartControl":{
         "process":"TRIED"
      }
}
```

En la consola de novedades del portal del Sistema de Control se informará cuando el control general sea reiniciado desde la API



Ejemplos:

```
query{
      restartControl(){}
}
```

stopControl

Detiene el control general

```
query{
      stopControl(){ }
}
```

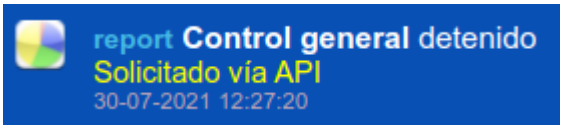
Detiene las funciones de control de acceso y control de ancho de banda

- parámetros
  - no requiere ningún parámetro
- respuesta **data**
  - *true* si se ejecutó el proceso y quedaron inactivos el control de acceso y el control de ancho de banda
  - *false* si no fue posible ejecutar el control o si a pesar de haberse ejecutado, por alguna condición del sistema o de la comunicación, el control general sigue inactivo
- respuesta **status**
  - **done** - confirma si se ejecutó el control [true|false]
  - **message** - informa qué sucedió al ejecutar la función de control
    - STOPPED: ejecutó la función de control y reinició el control general
    - FAILED: no pudo ejecutar la función de control
  - **control** - informa el estado en el que quedó el control general, más allá de si se ejecutó o no la función de control [ACTIVE|UNACTIVE]

- respuesta **errors**
  Repite el mismo valor de *message* dentro de la variable *process,* sólo cuando la ejecución falla.
  Ejemplo de respuesta *error*:

```
"errors":{
      "stopControl":{
         "process":"TRIED"
      }
}
```

En la consola de novedades del portal del Sistema de Control se informará cuando el control general sea detenido desde la API



Ejemplos:

```
query{
      restartControl(){}
}
```

statusControl

Obtiene el estado activo o inactivo del control general

```
query{
      statusControl(){ }
}
```

- parámetros
  - no requiere ningún parámetro
- respuesta **data**
  - *true* si está activo el control de ancho de banda
  - *false* si NO está activo el control de ancho de banda
- respuesta **status**
  - **done** - confirma si se ejecutó el control [true|false]
  - **control** - repite el valor de data pero en lugar de true|false utiliza ACTIVE|UNACTIVE

Ejemplos:

```
query{
      statusControl(){}
}
```

restartNetwork

Reinicia la configuración de la red

```
query{
      restartNetwork(){ }
}
```

Aplica las configuraciones de Redes del sistema, reiniciando también el firewall y la redirección de IP públicas si estos servicios están activos

- parámetros
  - no requiere ningún parámetro
- respuesta **data**
  - *true* si fue aplicada (reiniciada) la configuración de las redes
  - *false* si no fue reiniciada la configuración de las redes
- respuesta **status**
  - **done** - confirma si se ejecutó el control [true|false]

En la consola de novedades del portal del Sistema de Control se informará cuando las configuraciones de red sean reiniciadas desde la API

Ejemplos:

```
query{
      restartNetwork(){}
}
```

Sislander API client

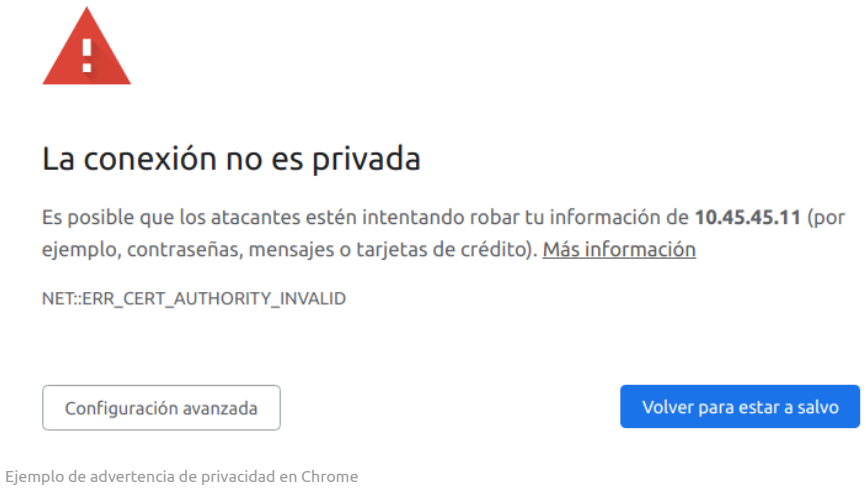
A partir de la versión 21.03 Sislander dispone también de un cliente en el servidor para consultar la API de ese mismo servidor.

Esto te permite probar fácilmente la ejecución de los comandos que luego enviarás desde otras aplicaciones/dispositivos.

¡Atención! Los comandos se ejecutan realmente en el sistema. Hay que tener mucho cuidado con las funciones de edición y borrado de datos.

Preparando el acceso al cliente API

1. Debes disponer de un operador con privilegios de *Acceso por API* configurado en el sistema de control del servidor. Si no lo has hecho, debes hacerlo como primer paso tal como se explica más arriba en la sección Autenticación.
2. ¡Atención! Si deseas acceder al cliente API por cualquiera de las redes WAN del servidor desde el lado WAN o desde internet debe estar abierto el puerto 443 en dicha red WAN como se explica en la sección Protocolo HTTPS.
3. La página con el cliente API se accede por https. En el primer acceso, en cada navegador que utilices, debes aceptar la excepción.
   Usando de *ejemplo* una IP *10.45.45.11* en el servidor, estos son los pasos (hay que reemplazar la IP en cada ejemplo por la IP pública o privada del servidor al cual accedes):
   a. Abrir en el navegador https://10.45.45.11/apiclient
   b. En la advertencia de seguridad hacer clic en *Configuración Avanzada* y luego clic en *Acceder a X.X.X.X (sitio no seguro)* o las opciones equivalentes según el navegador que se utilice.



Ejemplo de advertencia de privacidad en Chrome

Ejemplo de opción para omitir la advertencia de seguridad

Esta advertencia es normal en el acceso por web vía https a todo router o servidor excepto aquellos que tienen un dominio público asignado lo cual es raro que se implemente.
Puedes evitar esta advertencia y conectar en forma más segura al cliente API cargando en el navegador el certificado que puedes copiar en el sistema de control del servidor en la página *Operadores*, haciendo clic en la ayuda contextual que aparece junto a la opción *Acceso por API* .
En la mayoría de los casos suele ser más práctico simplemente “saltar" la advertencia.

Utilizando el cliente API

- En el campo más grande escribe o pega la consulta englobada en *query{}* o *mutation{}* como todas las consultas a la API de Sislander.
- En el campo *Operador API* ingresa el operador con privilegios de acceso API configurado en el servidor al cual accedes y en *Clave* la correspondiente contraseña.
- Si marcas *sólo respuesta API* no volverá a aparecer el formulario de consulta y deberás volver atrás o recargar para hacer una nueva consulta. La ventaja es que es más fácil copiar la respuesta. Es opcional.
- Clickeado en Query envías la **consulta real** al servidor, **modificando sus configuraciones** si es un query de edición. La respuesta aparecerá a la derecha del formulario de consulta (si has marcado *sólo respuesta API* aparecerá solo la respuesta).
- Puedes realizar una nueva consulta en la misma página de respuesta (si no marcaste *sólo respuesta API*).

MÁS EJEMPLOS

1. Agregar un cliente (usuario) con id #911

```
query{
	addClient(
		id:"911"
		name:"oneclient"
		password:"onepassword"
		firstName:"One"
		lastName:"Client"
		clientID:"23108054"
		phone:"+55 9558 4647"
		email:"oneclient@mymail.com"
		plan:"5"
	){
		id
		name
		password
		firstName
		lastName
		clientID
		phone
		email
		createdAt
		plan
	}
}
```

Respuesta


```
{
   "data":{
      "addClient":[{
         "id":"911",
         "name":"oneclient",
         "password":"*CE7E3864925E1A6574131F25ADD0AD6CE2CA41F6",
         "firstName":"One",
         "lastName":"Client",
         "clientID":null,
         "phone":"+55 9558 4647",
         "email":"oneclient@mymail.com",
         "createdAt":"2021-04-08",
         "plan":"5"
      }]
   },
   "status":{
      "addClient":{
         "id":"911",
         "done":true,
         "message":"ADDED"
      }
   }
}
```

2. Editar datos del cliente (usuario) con id #911

```
query{
      editClient(
            id:"911"
            name:"theclient"
            firstName:"Thierry"
            lastName:"Henry"
            email:"thenry@mymail.com"
            plan:"11"
      ){
            id
            name
            firstName
            lastName
            clientID
            phone
            email
            createdAt
            plan
      }
}
```

Respuesta


```
{
   "data":{
      "editClient":[{
         "id":"911",
         "name":"theclient",
         "firstName":"Thierry",
         "lastName":"Henry",
         "clientID":null,
         "phone":"+55 9558 4647",
         "email":"thenry@mymail.com",
         "createdAt":"2021-04-08",
         "plan":"11"
      }]
   },
   "status":{
      "editClient":{
         "id":"911",
         "done":true,
         "message":"UPDATED"
      }
   }
}
```

3. Obtener datos del cliente (usuario) con id #911

```
query{
      getClient(
            id:"911"
      ){
            id
            name
            password
            firstName
            lastName
            secondLastName
            clientID
            phone
            phone_2
            email
            createdAt
            plan
      }
}
```

Respuesta


```
{
   "data":{
      "getClient":[{
         "id":"911",
         "name":"theclient",
         "password":"*CE7E3864925E1A6574131F25ADD0AD6CE2CA41F6",
         "firstName":"Thierry",
         "lastName":"Henry",
         "secondLastName":null,
         "clientID":null,
         "phone":"+55 9558 4647",
         "phone_2":"",
         "email":"thenry@mymail.com",
         "createdAt":"2021-04-08",
         "plan":"11"
      }]
   },
   "status":{
      "getClient":{
         "id":"911",
         "done":true,
         "count":1
      }
   }
}
```

4. Obtener datos de los clientes (usuarios) asociados al plan (grupo) #8 con conexiones activas

```
query{
      getClient(
            plan:"8"
            status:true
      ){
            id
            name
            firstName
            lastName
            plan
      }
}
```

Respuesta


```
{
   "data":{
      "getClient":[{
         "id":"296",
         "name":"René Residente",
         "firstName":"",
         "lastName":"",
         "plan":"8"
      },
      {
         "id":"297",
         "name":"Vecindario de la Calle 13",
         "firstName":"",
         "lastName":"",
         "plan":"8"
      },
      {
         "id":"322",
         "name":"JULIO RODELO MENDOZA",
         "firstName":"",
         "lastName":"",
         "plan":"8"
      }]
   },
   "status":{
      "getClient":{
         "plan":"8",
         "status":true,
         "done":true,
         "count":3
      }
   }
}
```

4. Obtener datos de 2 usuarios en un mismo query

```
query{
      client1:getClient(id: "3") {
         id
         name
         plan
         firstName
         lastName
      }
      client2:getClient(id: "320") {
         name
         password
         firstName
         lastName
         secondLastName
         clientID
         phone
         email
      }
}
```

Respuesta


```
{
   "data":{
      "client1":[{
         "id":"3",
         "name":"THE GREATEST SAS OFICINA",
         "plan":"7",
         "firstName":"",
         "lastName":""
      }],
      "client2":[{
         "name":"LILIANA COLINA",
         "password":"*D9D3B88E6E054CB74F524706140471F5C9FF39A1",
         "firstName":"",
         "lastName":"",
         "secondLastName":"",
         "clientID":null,
         "phone":"",
         "email":""
      }]
   },
   "status":{
      "client1":{
         "id":"3",
         "done":true,
         "count":1
      },
      "client2":{
         "id":"320",
         "done":true,
         "count":1
      }
   }
}
```

6. Borrar cliente (usuario) con id #36

```
query{
      delClient(id:"36"){data result}
}
```

Respuesta


```
{
   "data":{
      "delClient":true
   },
   "status":{
      "delClient":{
         "id":"36",
         "done":true,
         "message":"DELETED"
      }
   }
}
```

7. Agregar una conexión por IP fija y periodo de inicio/fin al usuario con id #911

```
query{
      addClientDevice(
            id:"911"
            ip:"10.44.44.25"
            mac:"00:11:22:33:44:55"
            byDate:true
            start:"2021-09-01"
            end:"2021-09-30"
      ){
            id
            connection
            byLogin
            byLoginUnique
            ip
            mac
            publicIP
            lastModified
            byDate
            start
            end
            startAdvanced
            renew
            status
            byLapse
            lapseDays
            lapseHours
            lapseMinutes
            noDhcp
            logged
      }
}
```

Respuesta


```
{
   "data":{
      "addClientDevice":[{
         "id":"911",
         "connection":"771",
         "byLogin":"",
         "byLoginUnique":"",
         "ip":"10.44.44.25",
         "mac":"00:11:22:33:44:55",
         "publicIP":"",
         "lastModified":"2021-04-08 17:22:29",
         "byDate":"1",
         "start":"2023-09-01 00:00:00",
         "end":"2023-09-30 15:23:23",
         "startAdvanced":"",
```

```
            "renew":"",
            "status":{
               "status_enabled":true,
               "status_active":false,
               "status_unactive_reason":"FUTURE"
            },
            "byLapse":"",
            "lapseDays":"",
            "lapseHours":"",
            "lapseMinutes":"",
            "noDhcp":"",
            "logged":"0"
         }]
      },
      "status":{
         "addClientDevice":{
            "id":911,
            "done":true,
            "status":"ADDED"
         }
      }
   }
```

8. Agregar una conexión por login con un lapso de tiempo límite a partir del logueo

```
query{
	addClientDevice(
		id:"911"
		byLogin:true
		byLapse:true
		lapseDays:"5"
		lapseHours:"12"
	){
		id
		connection
		byLogin
		lastModified
		status
		byLapse
		lapseDays
		lapseHours
		lapseMinutes
		logged
	}
}
```

Respuesta


```
{
   "data":{
      "addClientDevice":[{
         "id":"911",
         "connection":"772",
         "byLogin":"1",
         "lastModified":"2021-04-08 17:26:08",
         "status":{
            "status_enabled":true,
            "status_active":false,
            "status_unactive_reason":"UNLOGGED"
         },
         "byLapse":"on",
         "lapseDays":"05",
         "lapseHours":"12",
         "lapseMinutes":"00",
         "logged":"0"
      }]
   },
   "status":{
      "addClientDevice":{
         "id":911,
         "done":true,
         "status":"ADDED"
      }
   }
}
```

9. Modificar la conexión recién agregada

```
query{
	editClientDevice(
		id:"911"
		connection:"772",
		byLoginUnique:true
		byLapse:true
		lapseDays:"7"
		lapseHours:"12"
		lapseMinutes:"30"
	){
		id
		connection
		byLogin
		byLoginUnique
		lastModified
		byLapse
		lapseDays
		lapseHours
		lapseMinutes
		logged
	}
}
```

Respuesta


```
{
   "data":{
      "editClientDevice":[{
         "id":"911",
         "connection":"772",
         "byLogin":"1",
         "byLoginUnique":"1",
         "lastModified":"2021-04-08 17:59:21",
         "byLapse":"on",
         "lapseDays":"07",
         "lapseHours":"12",
         "lapseMinutes":"30",
         "logged":"0"
      }]
   },
   "status":{
      "editClientDevice":{
         "id":911,
         "done":true,
         "status":"UPDATED"
      }
   }
}
```

10. Obtener datos de las conexiones de usuario #911

```
query{
      getClientDevices(id:"911"){
            id
            connection
            byLogin
            byLoginUnique
            ip
            mac
            publicIP
            createdAt
            byDate
            start
            end
            startAdvanced
            byDateRenew
            byLapse
            lapseDays
            lapseHours
            lapseMinutes
            noDhcp
            status
      }
}
```

Respuesta


```
{
   "data":{
      "getClientDevices":[{
         "id":"911",
         "connection":"772",
         "byLogin":true,
         "byLoginUnique":true,
         "ip":"",
         "mac":"",
         "publicIP":"",
         "createdAt":null,
         "byDate":false,
         "start":"0000-00-00 00:00:00",
         "end":"0000-00-00 00:00:00",
         "startAdvanced":false,
         "byDateRenew":null,
         "status":"unactive",
         "byLapse":true,
         "lapseDays":"07",
         "lapseHours":"12",
         "lapseMinutes":"30",
         "noDhcp":""
         "status":{
```

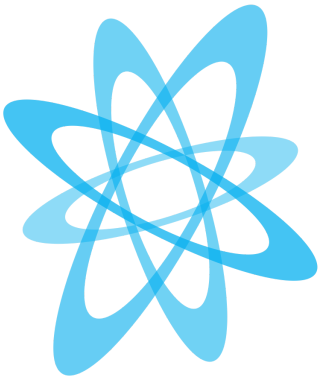
```
            "status_enabled":true,
            "status_active":false,
            "status_unactive_reason":"UNLOGGED"
          },
      },
      {
         "id":"911",
         "connection":"773",
         "byLogin":false,
         "byLoginUnique":false,
         "ip":"10.44.44.25",
         "mac":"00:11:22:33:44:55",
         "publicIP":"",
         "createdAt":null,
         "byDate":true,
         "start":"2021-01-01 00:00:00",
         "end":"2023-12-31 15:23:23",
         "startAdvanced":false,
         "byDateRenew":null,
         "status":"unactive",
         "byLapse":false,
         "lapseDays":"",
         "lapseHours":"",
         "lapseMinutes":"",
         "noDhcp":""
         "status":{
            "status_enabled":true,
            "status_active":true,
          },
      }]
   },
   "status":{
      "getClientDevices":{
         "id":911,
         "count":2,
         "done":true
      }
   }
}
```

Información Complementaria

Sobre el número de identificación tributaria/personal en los usuarios

Respecto al *Nro. único de identificación personal o tributaria* en los Usuarios, la ayuda de Sislander en la misma Edición de Usuarios del sistema de control ahora dice:

> **Número único de identificación personal o tributaria del usuario. [Opcional]**
>
> – En ocasiones es necesario identificar al Usuario por identificación legal además de por el ID del sistema.
>
> – El concepto de Usuario en Sislander se puede concebir y utilizar de 2 formas: como Usuario real (persona o empresa) o como un Servicio (también conocido como Contrato en algunos sistemas) con sus correspondientes Conexiones (dispositivos).
>
> – Si 2 o más usuarios tienen el mismo Número de identificación eso estaría indicando que la misma persona o empresa tiene 2 o más servicios.
>
> – Por ejemplo: Configuras 2 Usuarios, JuanPerez-Centro y JuanPerez-Norte, con el mismo Número de Identificación, lo cual indicaría que esos Usuarios son en realidad 2 Servicios de la persona o empresa Juan Pérez.
>
> – Para facilitar la tarea de los sistemas de administración y gestión conectados, en futuras versiones Sislander irá separando gradualmente el concepto de Usuario del concepto de Servicio.
>
> – Dado que en la mayoría de los casos hay un Usuario por Servicio, se han mantenido esos conceptos como uno sólo hasta ahora.



www.sislander.com
info@sislandsoft.com