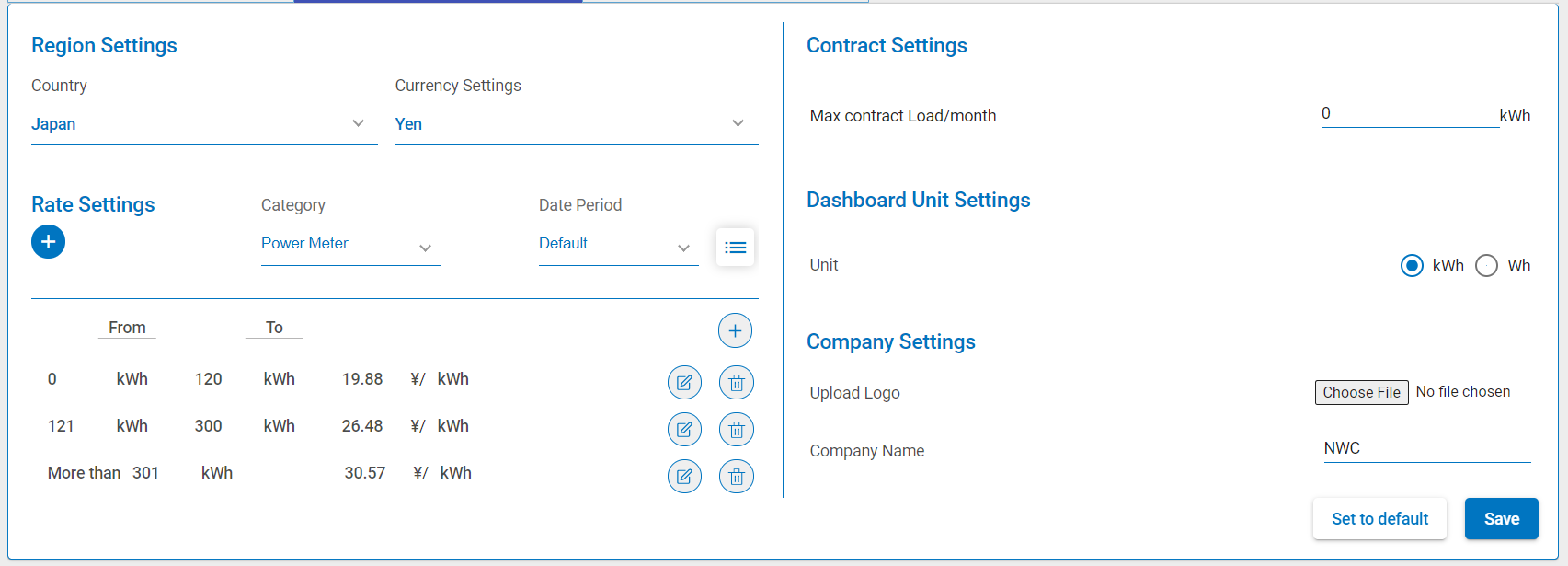
**/\* Application configuration \*/**

****

**/\*\***

**\* Save Configuration**

**\*/**

saveConfig() {

// Check if there are changes

const hasRateChanges = this.hasRateChanged();

const hasChanges = this.valuesChanged || hasRateChanges || this.isLogoUpdated;

if (!hasChanges) {

this.notificationService.showWarning(

this.authService.languageObject.common.warningMessage.noChanges,

this.authService.languageObject.settings.appConfig

);

return;

}

// validate rate first

if(!this.isPowerMeterCategory && this.selectedRate.value <= 0){

this.notificationService.showWarning(this.authService.languageObject.settings.appConfigScreen.warningMessage.incorrectRateValue,

this.authService.languageObject.settings.appConfig);

return false;

}

const valid = this.isPowerMeterCategory ? this.validateEconomicRange() : true;

if (valid) {

const data = {

configData: {

configuration: this.configuration

}

};

if (this.isLogoUpdated) {

this.apiService.uploadLogo(this.formData).subscribe((response) => {

this.form.get('file').setValue('');

this.uploadField.nativeElement.value = '';

this.isLogoUpdated = false;

this.authService.refreshHeaderLogo.next(true);

});

}

if (hasChanges) {

this.store.dispatch(new SaveConfiguration(data));

this.lastConfig = JSON.parse(JSON.stringify(this.configuration));

this.cancelEdit();

}

// Update Rate Settings

if (hasRateChanges) {

const requestParams: UpdateRateConfigRequest = {

id: this.selectedRate.\_id,

category: this.selectedRate.category,

rates: this.isPowerMeterCategory ? this.selectedRate.rates : this.selectedRate.value

};

this.store.dispatch(new UpdateDatePeriod(requestParams, !hasChanges));

this.cancelEdit();

}

this.valuesChanged = false;

// this.apiService.configDetails.activeLanguage = this.configuration.language.activeLanguage;

this.apiService.configDetails.companyName = this.configuration.companyDetails.name;

this.apiService.configDetails.activeUnit = this.configuration.energyUnits.activeUnit;

}

}

**/\*\***

**\* Check valid rate range**

**\*/**

validateEconomicRange(): boolean {

const ranges = this.selectedRate.rates;

ranges[0].rangeStart = Number(ranges[0].rangeStart);

if (ranges[0].rangeStart !== 0) {

this.notificationService.showWarning(

this.authService.languageObject.settings.appConfigScreen.warningMessage.startRangeZero,

this.authService.languageObject.settings.appConfig

);

return false;

}

let result = true;

let i = 0;

for (i = 0; i < ranges.length; i++) {

const recordNumber = i + 1;

let message: string

let rangeSuffix = (

recordNumber == 1 ?

this.authService.languageObject.settings.appConfigScreen.first :

recordNumber === 2 ? this.authService.languageObject.settings.appConfigScreen.second :

recordNumber === 3 ? this.authService.languageObject.settings.appConfigScreen.third :

recordNumber + this.authService.languageObject.settings.appConfigScreen.th

);

if (ranges[i].rangeStart > ranges[i].rangeEnd && i !== ranges.length - 1) {

message = this.authService.languageObject.settings.appConfigScreen.warningMessage.rangeEndValidation;

message = message.replace('VALUE',rangeSuffix);

this.notificationService.showWarning(message,this.authService.languageObject.settings.appConfig);

result = null;

break;

}

if (ranges[i - 1] && ranges[i].rangeStart !== ranges[i - 1].rangeEnd + 1) {

result = false;

break;

}

if (ranges[i + 1] && ranges[i].rangeEnd !== ranges[i + 1].rangeStart - 1) {

result = false;

break;

}

if (!ranges[i + 1] && (ranges[i].rangeEnd || ranges[i].rangeEnd === 0)) {

this.notificationService.showWarning(

this.authService.languageObject.settings.appConfigScreen.warningMessage.lastRangeBlank,

this.authService.languageObject.settings.appConfig

);

result = null;

break;

}

if(ranges[i].rangeStart == ranges[i].rangeEnd){

message = this.authService.languageObject.settings.appConfigScreen.warningMessage.rangeIncorrectValue;

message = message.replace('POSITION', rangeSuffix);

this.notificationService.showWarning(message, this.authService.languageObject.settings.appConfig);

return false;

}

if (ranges[i].rate <= 0) {

message = this.authService.languageObject.settings.appConfigScreen.warningMessage.rateIncorrectValue;

message = message.replace('POSITION', rangeSuffix);

this.notificationService.showWarning(message,this.authService.languageObject.settings.appConfig);

return false;

}

}

if (result == false) {

const recordNumber = i + 1;

let message: string = this.authService.languageObject.settings.appConfigScreen.warningMessage.ecoRangeIncorrect;

let rangeSuffix = (

recordNumber == 1 ?

this.authService.languageObject.settings.appConfigScreen.first :

recordNumber === 2 ? this.authService.languageObject.settings.appConfigScreen.second :

recordNumber === 3 ? this.authService.languageObject.settings.appConfigScreen.third :

recordNumber + this.authService.languageObject.settings.appConfigScreen.th

);

message = message.replace('VALUE', rangeSuffix);

rangeSuffix = (

recordNumber + 1 == 1

? this.authService.languageObject.settings.appConfigScreen.first

: recordNumber + 1 === 2

? this.authService.languageObject.settings.appConfigScreen.second

: recordNumber + 1 === 3

? this.authService.languageObject.settings.appConfigScreen.third

: recordNumber + 1 + this.authService.languageObject.settings.appConfigScreen.th);

message = message + this.authService.languageObject.settings.appConfigScreen.endmustbe + '' + rangeSuffix + ' ' + this.authService.languageObject.settings.appConfigScreen.range + this.authService.languageObject.settings.appConfigScreen.startless;

this.notificationService.showWarning(

message,

this.authService.languageObject.settings.appConfig

);

}

return result;

}