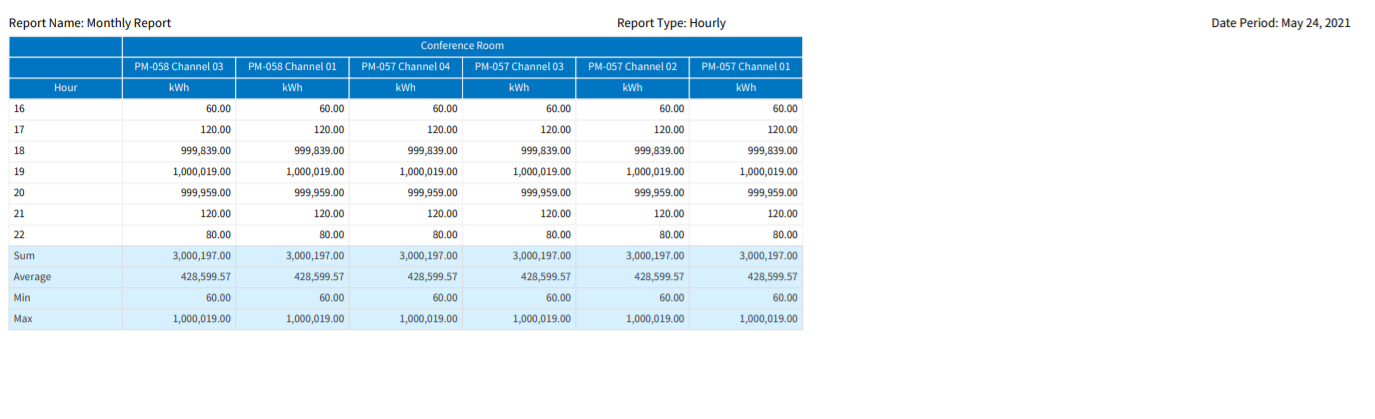
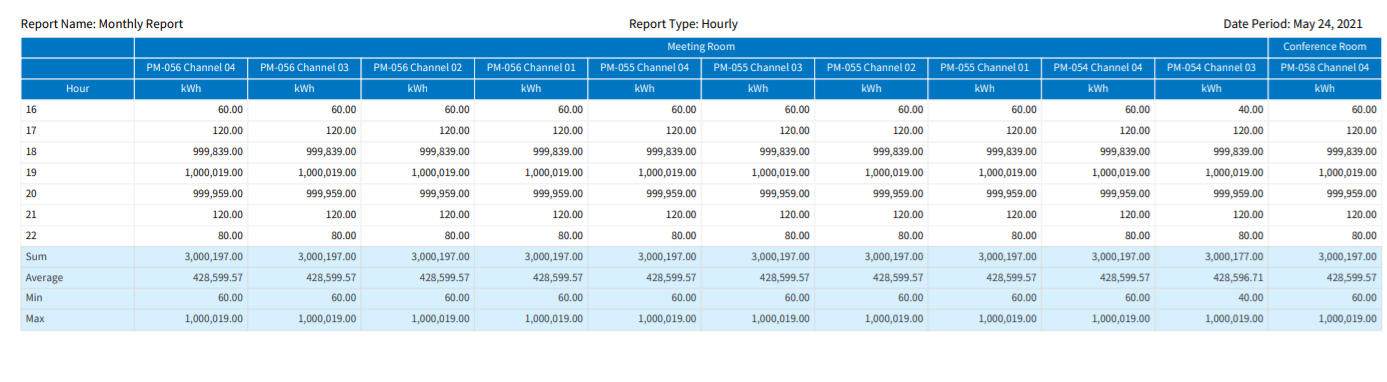
**/\* Generate multiple column PDF \*/**

****

exportTOPdf() {

// JSPDF Doc Initialization

let doc = new jsPDF({

format: 'a4',

unit: 'mm',

orientation: ["l"],

pagesplit: true

});

// Added font for handling Japanese Text

doc.addFileToVFS('NotoSansCJKjp-Regular.ttf', JapaneseFont);

const horizontalMargin = 5.5;

const verticalMargin = 10;

const pageWidth = doc.internal.pageSize.getWidth();

const pageHeight = doc.internal.pageSize.getHeight();

const contentWidth = pageWidth - horizontalMargin;

const contentHeight = pageHeight - verticalMargin;

const maxColumnsPerPage = 12;

const maxRowsPerPage = 31; // 3 -> headers, 15 -> body, 4 -> Footer

const totalReportColumns = this.displayedColumns.length - 1;

const numberOfPagesForOneTable = Math.ceil(totalReportColumns / (maxColumnsPerPage - 1));

// Table Data as Array Extraction Logic

let mainDataArr = [];

if (this.reportType == ConfigReportType.Daily) {

mainDataArr = this.dataSource.map((row, i) => {

return this.displayedColumns.map(dc => {

const value = row[dc.key] ? row[dc.key] : '';

if (row.class) return { content: value, styles: { fillColor: "#e8e6e6" } };

return { content: value };

});

});

} else {

mainDataArr = this.dataSource.map(row => {

return this.displayedColumns.map(dc => row[dc.key]);

});

}

// Header Data as Array Extraction Logic

let groupHeaders = [];

let dataPointHeaders = [];

let unitHeaders = [];

this.displayedColumns.map((x, i) => {

groupHeaders.push(x.groupName);

dataPointHeaders.push(x.dataPointName);

unitHeaders.push(x.value);

});

// Footer Data as Array Extraction Logic

const footerArr = this.footerData.map(footer => footer.map(f => f.value));

// Adding first column values after every N of columns

for(let i = 1; i < numberOfPagesForOneTable; i++) {

groupHeaders.splice(i \* maxColumnsPerPage, 0, groupHeaders[0]);

dataPointHeaders.splice(i \* maxColumnsPerPage, 0, dataPointHeaders[0]);

unitHeaders.splice(i \* maxColumnsPerPage, 0, unitHeaders[0]);

footerArr.forEach((footer, fIndex) => {

footer.splice(i \* maxColumnsPerPage, 0, footer[0]);

});

mainDataArr.forEach((row, rIndex) => {

row.splice(i \* maxColumnsPerPage, 0, row[0]);

});

}

// Making Chunks for Headers

let groupHeadersChunks = chunk(groupHeaders, maxColumnsPerPage).map(header => {

const headerObj = {};

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

headerObj[header[i]] = 1 + (headerObj[header[i]] || 0);

}

return Object.keys(headerObj).map(key => {

return { content: key, colSpan: headerObj[key] };

});

});

const dataPointHeadersChunks = chunk(dataPointHeaders, maxColumnsPerPage);

const unitHeadersChunks = chunk(unitHeaders, maxColumnsPerPage);

// Making Chunks for Footers

const footerDataArr = footerArr.map(footer => chunk(footer, maxColumnsPerPage));

// Making Chunks for Data

const chunks = chunk(mainDataArr, maxRowsPerPage);

let pageNumber = 0;

let totalPages = chunks.length \* numberOfPagesForOneTable;

var cellWidth = Math.floor(contentWidth / maxColumnsPerPage);

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

const data = chunks[i];

var tableDataChunks = data.map(row => {

return chunk(row, maxColumnsPerPage);

});

let dateLabel = this.authService?.languageObject?.configReport.datePeriod;

for (let j = 0; j < numberOfPagesForOneTable; j++) {

const headersData = [groupHeadersChunks[j], dataPointHeadersChunks[j], unitHeadersChunks[j]];

const footerData = footerDataArr.map(d => d[j]);

const tableData = tableDataChunks.map(d => d[j]);

doc.addFont('NotoSansCJKjp-Regular.ttf', 'NotoSansCJKjp', 'normal');

doc.setFont('NotoSansCJKjp');

doc.autoTable({

body: tableData,

head: headersData,

didDrawPage: (data) => {

let pageInfo = doc.internal.getCurrentPageInfo();

doc.setFontSize(7);

doc.text(`${this.languageObject?.configReport.reportName}: ${this.reportName}`, horizontalMargin, 8, 'left');

doc.text(`${this.languageObject?.configReport.reportType}: ${this.reportTypeLabel}`, contentWidth/2, 8, 'center');

doc.text(`${dateLabel}: ${this.datePeriod}`, contentWidth - 2, 8, 'right');

doc.text(this.languageObject?.common.page + ' ' + pageInfo.pageNumber, contentWidth/2, pageHeight - 5, 'center');

},

foot: footerData,

margin: { vertical: verticalMargin, horizontal: horizontalMargin },

tableWidth: 'wrap',

theme: 'plain',

columnStyles: {

cellWidth: 'auto',

},

styles: {

overflow: 'linebreak',

cellWidth: cellWidth,

font: 'NotoSansCJKjp',

},

headStyles: {

fontSize: 6,

halign: 'center',

lineWidth: 0.2,

fillColor: '#0075c1',

textColor: '#FFFFFF',

lineColor: "#f2f2f2",

rowSpan: 1,

fontStyle: 'bold',

valign: 'middle',

cellPadding: { top: 1, bottom: 1, left: 1, right: 1 }

},

bodyStyles: {

fontSize: 6,

halign: 'right',

valign: 'middle',

lineWidth: 0.2,

lineColor: "#f2f2f2",

cellPadding: { top: 1, bottom: 1, left: 1, right: 1 }

},

footStyles: {

halign: 'right',

fontSize: 6,

textColor: '#404040',

fillColor: '#D6EFFF',

lineWidth: 0.2,

lineColor: "#ddd",

fontStyle: 'bold',

valign: 'middle',

cellPadding: { top: 1, bottom: 1, left: 1, right: 1 }

},

didParseCell: (data) => {

if (data.cell.section == "body" && data.column.index === 0) {

// data.cell.styles.textColor = '#FFFFFF';

data.cell.styles.halign = 'left';

}

if (data.cell.section == "foot" && data.column.index === 0) {

data.cell.styles.halign = 'left';

}

}

});

pageNumber++;

if (pageNumber < totalPages) doc.addPage();

}

}

// Open PDF document in new tab

// window.open(doc.output('bloburl'), '\_blank');

// Download PDF document

doc.save(`${this.exportFileName}.pdf`);

}