Displays Images Reference

Insert and manage images in displays.

Displays Images (Reference) rovides a centralized repository for image resources used throughout the solution, stored within the project database for backgrounds, icons, and graphical elements.

Display Images provides:

- Embedded image storage
- Multiple format support
- Dynamic brush creation
- Resource replacement
- Fill pattern usage
- Runtime color dynamics

Once imported, images become part of the solution database, eliminating dependency on external files.

Supported Formats
Importing Images
Using Images
Image Properties
Image Management
Dynamic Image Usage
Best Practices
Performance Considerations
Common Applications
Troubleshooting
Advanced Features

Supported Formats

Format	Extension	Use Case
Bitmap	.bmp	Uncompressed images
GIF	.gif	Simple animations
Icon	.ico	Application icons
JPEG	.jpg	Photographs
SVG	.svg	Scalable vectors
PNG	.png	Transparent graphics
TIFF	.tiff	High-quality images
WDP	.wdp	HD Photo format

Importing Images

Import Process

1. Navigate to **Displays Images**
2. Click **Import Images**
3. Select image files
4. Click **Open**

Images are embedded in the project database.

Batch Import

- Select multiple files
- Maintain folder structure
- Auto-generate names
- Preview before import

Using Images

In Display Editor

1. Open **Displays Draw**
2. Select **Images** category in Components Panel
3. Drag image to canvas
4. Configure stretch mode

As Object Fill

Apply images as brushes:

- Rectangles
- Ellipses
- Paths
- Backgrounds
- LabelBoxes

Stretch Modes

Mode	Description	Use Case
None	Original size	Icons, logos
Fill	Fill container	Backgrounds
Uniform	Maintain aspect ratio	Photos
UniformToFill	Fill and crop	Wallpapers

Image Properties

Property	Description	Editable
Name	Resource identifier	Yes
Description	Image documentation	Yes
Folder	Organization path	Yes
ResourceType	File type	Read-only
Dimension	Width x Height	Read-only
Size	File size in bytes	Read-only

Image Management

Replacing Images

1. Keep same resource name
2. Import new version
3. Auto-updates throughout solution
4. No display changes needed

Organizing Images

```
/Backgrounds
  /Screens
  /Popups
/Icons
  /Navigation
  /Equipment
/Logos
  /Company
  /Products
```

Dynamic Image Usage

Runtime Image Change

csharp

```csharp
// Change image source
@Display.MyImage.Source = "NewImageName";

// Toggle between images
if (condition)
    @Display.Background.Fill = "Image1";
else
    @Display.Background.Fill = "Image2";
```

Image as Color

csharp

```csharp
// Use image in color dynamics
@Display.Rectangle1.Fill = "ImageBrush:Logo";

// Conditional image fill
@Display.Shape.Fill = @Tag.State == 1 ?
    "ImageBrush:ActiveImage" :
    "ImageBrush:InactiveImage";
```

Best Practices

1. **Optimize Size** - Compress before import
2. **Use Appropriate Formats** - PNG for transparency, JPG for photos
3. **Organize Folders** - Logical structure
4. **Name Consistently** - Clear identifiers
5. **Document Purpose** - Use descriptions
6. **Consider Resolution** - Match display requirements
7. **Test Scaling** - Verify stretch modes

Performance Considerations

Image Optimization

- Limit file sizes
- Use appropriate resolution
- Avoid oversized images
- Consider vector formats

Memory Management

- Images load on demand
- Cached after first use
- Released when not visible
- Monitor total size

Common Applications

Backgrounds

xml

```xml
<Page Background="ImageBrush:PlantLayout">
  <!-- Display content -->
</Page>
```

Equipment States

csharp

```csharp
// Motor image based on state
string motorImage = @Tag.Motor.Running ?
    "Motor_Running" : "Motor_Stopped";
@Display.MotorImage.Source = motorImage;
```

Navigation Icons

xml

```xml
<Button>
  <Image Source="Icon_Home" Width="32" Height="32"/>
</Button>
```

Troubleshooting

Image not displaying:

- Verify name spelling
- Check import success
- Review stretch mode
- Test in runtime

Poor quality:

- Use higher resolution
- Try different format
- Check stretch settings
- Avoid excessive scaling

Memory issues:

- Reduce image sizes
- Limit simultaneous images
- Use image compression
- Monitor resource usage

Update not reflecting:

- Maintain same name
- Rebuild solution
- Clear cache
- Restart runtime

Advanced Features

Image Libraries

Create reusable image sets:

csharp

```csharp
public class ImageLibrary
{
    public static string GetStateImage(int state)
    {
        return state switch
        {
            0 => "State_Idle",
            1 => "State_Running",
            2 => "State_Alarm",
            _ => "State_Unknown"
        };
    }
}
```

Dynamic Loading

csharp

```csharp
// Load image from database
byte[] imageData = @Dataset.Query.Images.GetImage(id);
@Display.DynamicImage.LoadFromBytes(imageData);
```

In this section...