**Junkers & Muellers GmbH QR Code Description**

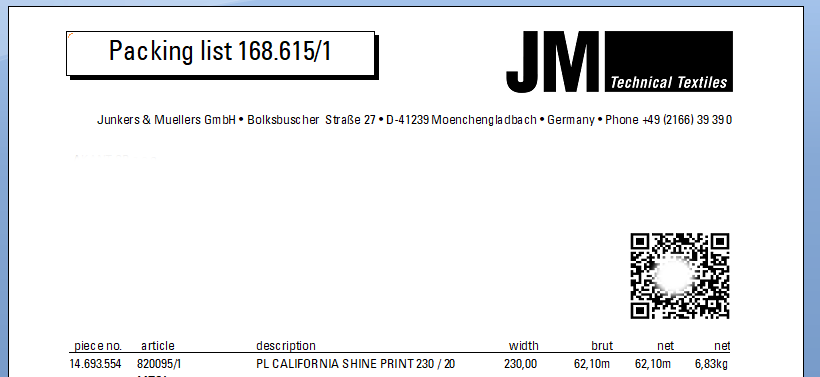
At JM, we continuously strive to enhance communication with our customers by implementing technical solutions that maximize the value of our business relationships.

We are pleased to introduce 2D QR codes (Quick Response codes) on all packing lists and item labels. Each QR code contains a link to a JM web server that returns the relevant data in XML format. Due to data volume constraints, the information cannot be stored directly within the QR code itself. A typical piece record contains approximately 2,000 bytes of data, while the maximum capacity of a QR code is approximately 4,000 alphanumeric characters (bytes). Storing this amount of data directly would result in impractically large QR codes on piece labels. Additionally, this server-based approach enables us to respond quickly to customer requests without modifying the physical labels.

QR Code Types

JM utilizes four types of QR codes, each serving a specific purpose:

* Packing list : [http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetPL?uid=[uid](http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetPL?uid=%5buid)]

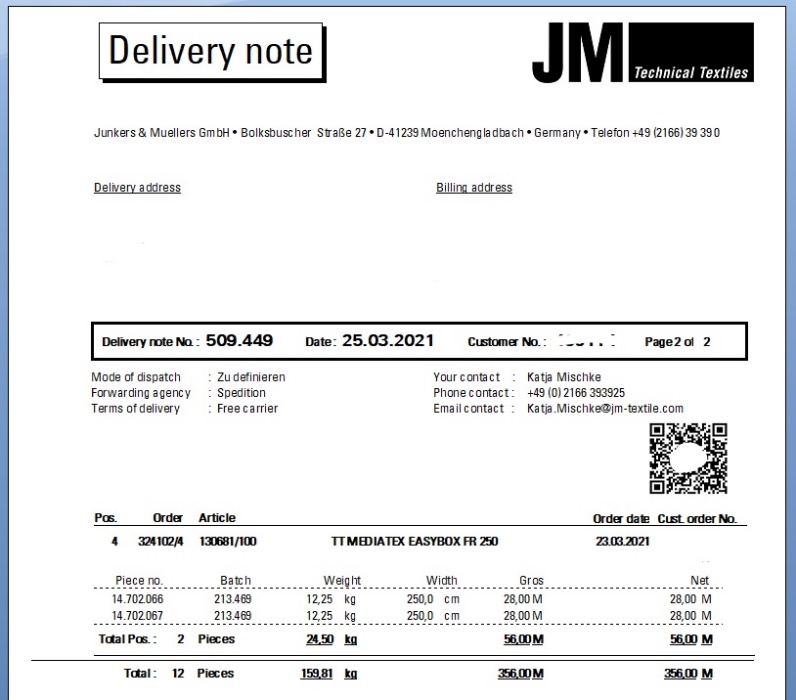


(Example QR code not scannable to protect customer data)

* Shipping Unit : [http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetSU?uid=[uid]&su=[su](http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetSU?uid=%5buid%5d&su=%5bsu)]

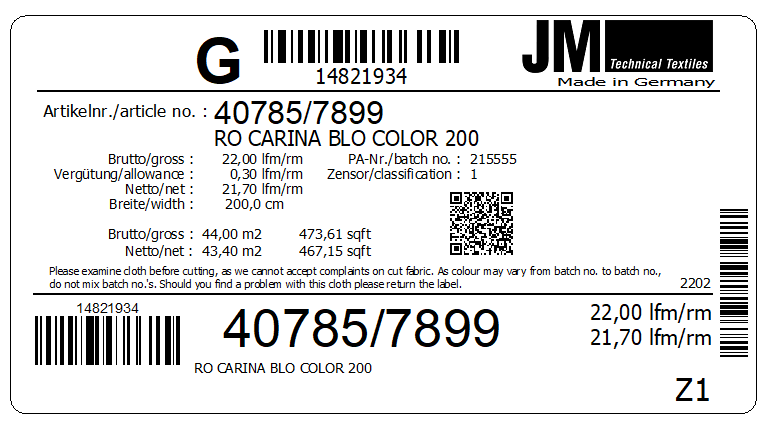


* Delivery note [http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetDN?uid=[uid](http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetDN?uid=%5buid)]



(Example QR code not scannable to protect customer data)

* Piece label : [http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetPiece?uid=[uid](http://ws.jmtt.eu/GetPieceData/Getpiecedata.asmx/GetPiece?uid=%5buid)]



**Security Note:** The [uid] parameter contains a unique identifier that ensures data security by preventing unauthorized access to other customers' information.

# Developer Resources

JM provides a comprehensive sample application written in C# for Windows, complete with a full Visual Studio solution. The application is available for download from GitHub:

https://github.com/JunkersAndMuellers/JM-QR-Code-Data-Example

This application serves multiple purposes:

* Provides a fully functional testing environment
* Includes all necessary data structures for developing customer-specific applications
* Can be directly integrated into C# customer applications
* Compatible with Xamarin for Android development
* Serves as a reference for implementations in other languages (C++, Java, etc.), with appropriate language-specific adaptations

The packing list data have the following structure:

public class PLRecord

{

public int PackinglistNo { get; set; }

public string CustomerName { get; set; }

public string CustomerStreet { get; set; }

public string CustomerCountry { get; set; }

public string CustomerZipcode { get; set; }

public string CustomerCity { get; set; }

public int PieceCount { get; set; }

public double TotalWeight { get; set; }

public int ShippingUnitCount { get; set; }

public List<ShippingUnitRecord> ShippingUnits { get; set; }

}

The <ShippingUnitRecord> list have the following structure:

public class ShippingUnitRecord

{

public int PackinglistNo { get; set; }

public int ShippingUnitNo { get; set; }

public string ShippingUnitDescription { get; set; }

public double TotalWeight { get; set; }

public int PieceCount { get; set; }

public List<PieceRecord> Pieces { get; set; }

}

The delivery note record have the following structure:

public class DNRecord

{

public int DeliveryNoteNo { get; set; }

public string CustomerName { get; set; }

public string CustomerStreet { get; set; }

public string CustomerCountry { get; set; }

public string CustomerZipcode { get; set; }

public string CustomerCity { get; set; }

public string DeliveryName { get; set; }

public string DeliveryStreet { get; set; }

public string DeliveryCountry { get; set; }

public string DeliveryZipcode { get; set; }

public string DeliveryCity { get; set; }

public string ContactName { get; set; }

public string ContactPhone { get; set; }

public string ContactMail { get; set; }

public int PieceCount { get; set; }

public double TotalWeight { get; set; }

public List<PieceRecord> Pieces { get; set; }

}

The <PieceRecord> list have the following structure:

public class PieceRecord

{

public int PieceNo { get; set; }

public int BatchNo { get; set; }

public string ArticleNo { get; set; }

public string ArticleDescription { get; set; }

public int? MachineID { get; set; }

public decimal? Brut { get; set; }

public decimal? Allowance { get; set; }

public decimal? Net { get; set; }

public decimal? BrutM2 { get; set; }

public decimal? AllowanceM2 { get; set; }

public decimal? NetM2 { get; set; }

public decimal? BrutCell { get; set; }

public decimal? AllowanceCell { get; set; }

public decimal? NetCell { get; set; }

public decimal? CellCount { get; set; }

public decimal? WeightBrut { get; set; }

public decimal? Tare { get; set; }

public decimal? WeightNet { get; set; }

public decimal? lfdm { get; set; }

public decimal? Witdh { get; set; }

public string Quality { get; set; }

public decimal? WeightM2 { get; set; }

public decimal? BrutFeet { get; set; }

public decimal? AllowanceFeet { get; set; }

public decimal? NetFeet { get; set; }

public decimal? BrutSquareFeet { get; set; }

public decimal? AllowanceSquareFeet { get; set; }

public decimal? NetSquareFeet { get; set; }

public decimal? BrutInch { get; set; }

public decimal? AllowanceInch { get; set; }

public decimal? NetInch { get; set; }

public decimal? BrutSquareInch { get; set; }

public decimal? AllowanceSquareInch { get; set; }

public decimal? NetSquareInch { get; set; }

public int? Parts { get; set; }

public string UID { get; set; }

public int? OrderNo { get; set; }

public int? OrderPos { get; set; }

public string PurchaseOrder { get; set; }

public string CustomerReferenz { get; set; }

public string CustomerArticleNo { get; set; }

public string CustomerArticleDesc { get; set; }

public int? DeliveryNoteNo { get; set; }

public int? DeliveryNotePos { get; set; }

public int? InvoiceNo { get; set; }

public string HSCode { get; set; }

public List<PieceErrorRecord> Errors { get; set; }

}

The < PieceErrorRecord> list have the following structure:

public class PieceErrorRecord

{

public string ErrorType { get; set; }

public double Position { get; set; }

public double Length { get; set; }

public int ErrorCode { get; set; }

}

**Units of Measurement**

All measurements follow the metric system unless explicitly specified otherwise:

* **Length values** (Net, Brut, Allowance, etc.): Metric units
* **Weight values** (WeightBrut, WeightNet, etc.): Metric units
* **Imperial units:** Clearly indicated in property names (e.g., BrutInch, NetSquareFeet)

**Technical Support**

For technical questions or support, please contact:

**it@jm-textile.com**