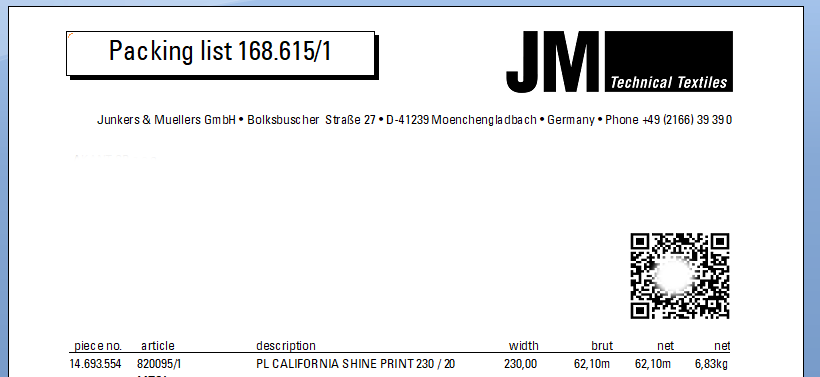
**Junkers & Muellers GmbH QR Code Description**

In order to improve communication with our customers, JM constantly strives to offer our customers technical solutions that enable them to get more out of our business relationship.

That is why JM is now offering a 2D QR code (quick response code) on all packing lists and item labels. The QR code contains a link to one of the JM web servers, which returns the relevant data in XML format. It is not possible to store the data directly in the QR code due to the amount of data. A piece record contains around 2000 bytes of data. The maximum amount of data that could be stored in a QR code would be around 4000 alphanumeric characters (bytes). The resulting size of the QR code would not be practical on piece labels. In addition, JM has the ability to react quickly to customer requests at any time without having to change anything on the label.

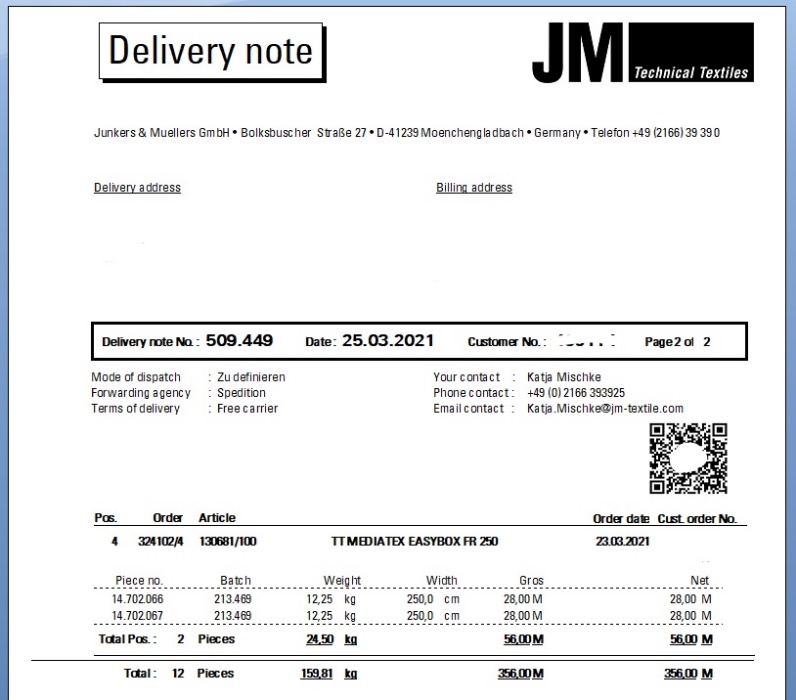
There are 3 types of QR codes JM is using. One on the packing list, one on the delivery note and one on the piece label. The links in the QR Codes are in the following format :

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



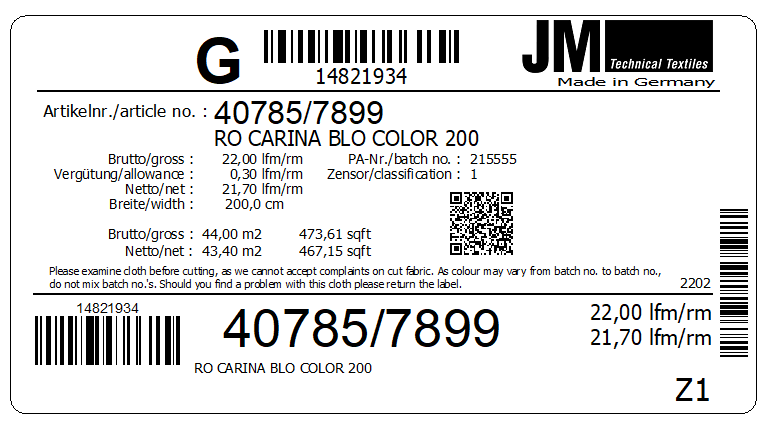
(Example QR code not scannable to protect customer data)

* 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)]



The [uid] contains a unique identifier, because of security reasons, to prevent that one customer can get data from

another.

Furthermore, JM provides a free sample program written in C# for Windows and contains a complete Visual Studio solution. It can be downloaded from Github (<https://github.com/AxelStallknecht/JM-QR-Code-Data-Example>).

This app does not only provide a fully functioning test, but also contains all the data structures necessary for developing customer-specific applications. Direct use in C # customer applications is just as possible as an application under Xamarin for Android. Appropriate adjustments must be made for customer applications in other languages such as C ++ or Java.

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 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; }

}

All values such as lengths (Net, Brut, Allowance ...) or weights (WeightBrut, WeightNet ....) correspond to the metric system, unless another unit is expressly mentioned (BrutInch, NetSquareFeet).

For any technical questions contact [edv@jm-textile.com](mailto:edv@jm-textile.com).