-62-

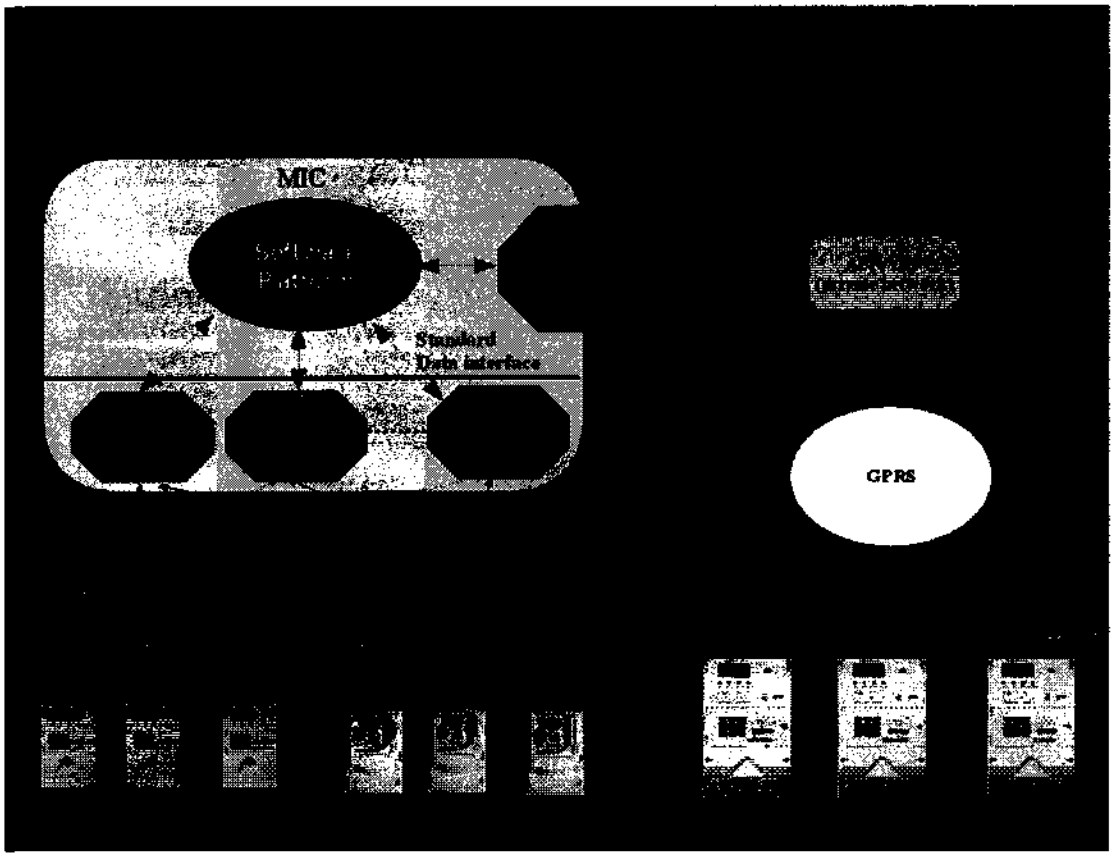
PART-2

(Unified Prepayment System API)

Table of Contents

1 Multi-factories Access Platform

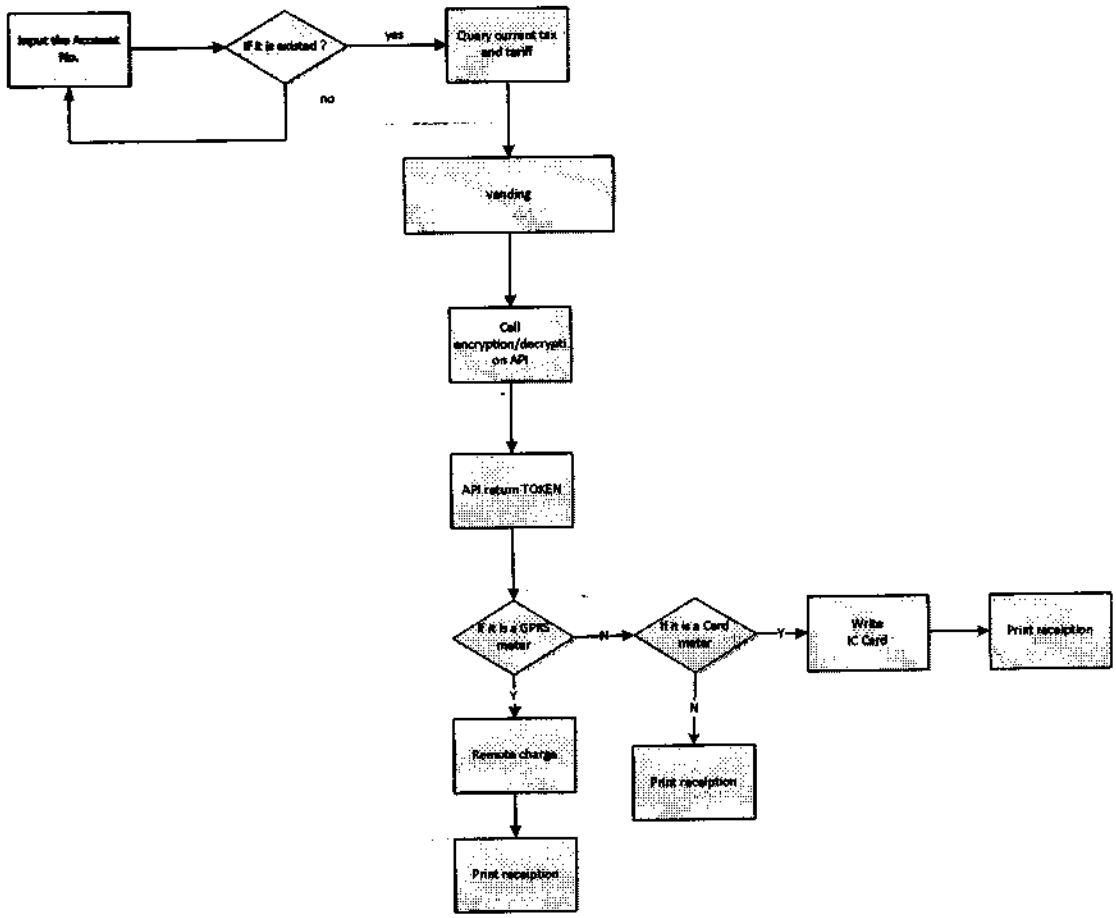
1.1 Access Platform Architecture Design



Each manufacturer provides its own encryption / decryption API, integrated into the system software.
When dealing with business based on different meters from each manufacturers, the system automatically calls the corresponding manufacturers' encryption / decryption API.
The system defines uniformly data exchange interface corresponding to different business ,such as input and output parameters.
When dealing with a business, for the keypad meter, the system will call the encryption / decryption API, generates the corresponding Token, and print ; for smart card meter, the system will call the encryption / decryption API, write the corresponding Token to encrypted data area of the smart card.
The encryption/decryption API from smart card manufacture must be able to generate Token which must contain the information of business type and length information. So that when smart card meter read the encrypted data in the card , the meter can separate the Token and identify the Token business itself.
POS read the smart cards from different manufactures via standard card read devices.
MIC communicates with GPRS meters of other factories according to standard data exchange mechanism .
Note: Token in this document means encrypted data.

2. Business Process

2.1 Vending as example:



3. Keypad Meters and Card Meter Access design

3.1 Error Code definition:

0 or empty: success

1: sequence number is not excepted

2: tariff Index argument error, not in scope

3: keyVersion argument not in scope

4: keyExpiredTime not in scope(0-255)

5: keyNo exceed 65535 or < 0

6: meterNo is not excepted

7: credit amount <= 0

8: unknown reason

3.2 Standard Interface for Generating A Variety of Token

3.2.1 Token Standard Interface Define

Programming language	JAVA
Functionality	Token standard interface class
Package	com.hx.ami.spi
Interface	Token

Class definition	public interface Token
	{
	Generatecredit Token
	public String getCreditToken(String meterNo, String sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, double amount);
	
	Generate key change Token
	public String getKeyChangeToken (String meterNo, String sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, String nSgc, int nTariffIndex, int nKeyVersion,int nKeyExpiredTime, long nKeyNo);
	
	Generate Management Token
	Generate clear balance Token
	public String getClearBalanceToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo);
	Generate clear event Token
	public String getClearEventToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo);
	Generate max Power Limit Mode Token
	public String getMaxPowerLimitToken(String meterNo,String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int activationModel, String date,int[] maxPowerLimits, int[] hours);
	Generate single Tariff Token
	public String getSingleTariffToken(String meterNo, String Sgc, int tarrifIndex,int keyVersion, int keyExpiredTime, long keyNo, int seqNo,String activatingDate, int activatingModel, int validDate, int rate);
	Generate step Tariff Token
	public String getStepTariffToken(String meterNo, String Sgc, int tarrifIndex,int keyVersion, int keyExpiredTime, long keyNo, int seqNo,String activatingDate, String validDate, int[] rates, int[] steps);
	Generate Tou Tariff Token
	public String getTOUTariffToken(String meterNo, String Sgc, int tarrifIndex,int keyVersion, int keyExpiredTime, long keyNo, int seqNo,String activatingDate, int activatingModel, int validDate, int[]

	rates, int[] times);
	Generate Friend Mode Token
	public String getFriendModeToken(String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int friendMode, int[] times, int[] days);
	Generate Public Holiday Token
	generateHolidayModeToken(String meterNo,String sgcId, int ti, int kv, int ke, int seq, int keyNo,int holidayMode, String[] days);
	Generate Change Meter Mode Token
	public String getChangeMeterModeToken(String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo,int mode);
	Generate Set Credit Amount Limit And Overdraw Amount Limit Token
	public String getSetCreditAmountLimitOrOverdrawAmountLimitToken(String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int amountType, int amountLimit);
	
	Generate Set Low Credit Warning Limit Token
	public String SetLowCreditWarningLimitToken(
	String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int amountType, int amountLimit);
	
	Generate LogoffReturn Token
	
	public String generateLogoffReturnToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, int keyNo, int seqNo);
	
	Resolve the Return Token
	public String resolveReturnToke(String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, String Token);
	
	4) Generate **Test Token**
	public String getTestToken (int manufacturingID, int control);
	}

3.2.2 Charge Interface Definition

Functionality	To get credit Token

Application	To call the function and generate credit token when vending. seqNo : Each token has sequence number except change-key-token. seqNo will increase 1 for each token generated and change to 1 when above 200; keyNo: when seqNo change from 200 to 1,keyNo add 1. keyExpiredTime: Global system parameter used as cipher fact. System can change it. Nothing to do with key expiration. Just a encryption fact. Key version: Global encryption fact, it will be changed in case all meter need to change keys.
Function name	**getCreditToken**
Parameter list	

[illegible]	[illegible]	[illegible]	[illegible]
meterNo	String	String of 12 numbers	Meter number
Sgc	String	String of 6 numbers	Supply group code System parameter.
tarrifIndex	Int	1-99	Tariff index When custom change tariff, this value may change to any of 1-99
keyVersion	Int	1-9	Key version Global encryption fact, it will be changed in case all meter need to change keys.
keyExpiredTime	Int	0-255	Global system parameter used as cipher fact. System can change it. Nothing to do with key expiration. Just a encryption fact
keyNo	Long	0-65535	Key Sequence, similar to key changed times, see above method descpt.

	seqNo	Int	1-200	Token Sequence
	amount	Int	0-99999999(scale&unit 0.01TK)	Credit amount
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> Note: errorCode see 3. Chart definition. If succeeded, need return an array of <tokens>elements.			

Function definition	public String getCreditToken(String meterNo, String sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int amount)
	{
	...
	}			

3.2.3 Key Change Interface Definition

Functionality	To get key change Token			
Application	As soon as user wants to charge money , to generate the manage token and test token ,and judge if the encryption factor has been changed, if changed , to call function and generate change token. This function is called at four case: Meter installed and custom purchase energy at first time Called if tariff changed SeqNo change from 200 to 1. Called when operator press menu to change key. For the first time registration, parameter defines as below: tariffIndex=0; keyVersion=0; keyExpiredTime=0; keyNo=0;			
Function name	**getKeyChangeToken**			
Parameter list	Parameter Name	Type	Range of Value	Description
	meterNo	String	String of 12 numbers	Meterno.

	Sgc	String	String of 6 numbers	Old Supply group code
	tariffIndex	Int	1-99	Old Tariff index
	keyVersion	Int	1-9	Old Key version
	keyExpiredTime	Int	0-255	Old Key expire time
	keyNo	Long	0-65535	Old Key Sequence
	nSgc	Int	String of 6 numbers	Current supply grop code
	nTariffIndex	Int	1-99	Current Tariff index
	nKeyVersion	Int	1-9	Current Key version
	nKeyExpiredTime	Int	0-255	Current Key expire time
	nKeyNo	Int	0-65535	Current Key Sequence

Return value

Return XML String

Xml format:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
        <errorCode></errorCode>
        <tokens>
                <token></token>
                ...
                <token></token>
        <tokens>
</result>
```

Note:

errorCode definition refer to chart 3.

If succeeded, need return an array of <tokens>elements

Function definition

```
public String getKeyChangeToken (String meterNo, String sgc, int
tarrifIndex,
int keyVersion, int keyExpiredTime, long keyNo, String nSgc, int
nTariffIndex, int nKeyVersion,int nKeyExpiredTime, int nKeyNo)
{
   ...
}
```

3.2.4 Management Token

3.2.4.1 Generate Clear Meter Balance Token

<table>
<tr><td>Functionality</td><td colspan="4">To get clear balance Token</td></tr>
<tr><td>Application</td><td colspan="4">To clear balance</td></tr>
<tr><td>Function name</td><td colspan="4">getClearBalanceToken</td></tr>
<tr><td>Parameter list</td><td colspan="4"><table>
<tr><th>Parameter name</th><th>Type</th><th>Scope or length</th><th>Description</th></tr>
<tr><td>meterNo</td><td>String</td><td>String of 12 numbers</td><td>Meterno.</td></tr>
<tr><td>sgc</td><td>String</td><td>String of 6 numbers</td><td>Supply group code</td></tr>
<tr><td>tariffIndex</td><td>Int</td><td>1-99</td><td>Tariff index</td></tr>
<tr><td>keyVersion</td><td>int</td><td>1-9</td><td>Key version</td></tr>
<tr><td>keyExpiredTime</td><td>int</td><td>0-255</td><td>Key expire time</td></tr>
<tr><td>keyNo</td><td>long</td><td>0-65535</td><td>Key Sequence</td></tr>
<tr><td>seqNo</td><td>int</td><td>1-255</td><td>Token Sequence</td></tr>
</table></td></tr>
<tr><td>Return value</td><td colspan="4">Return XML String
Xml format:
<?xml version="1.0" encoding="UTF-8"?>
<result>
<errorCode></errorCode>
<tokens></td></tr>
</table>

	<token></token> ... <token></token> <tokens> </result> Note: errorCode definition refer to chart 3. If succeeded, need return an array of <tokens>elements
Function definition	**public String getClearBalanceToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo) { ... }**

3.2.4.2 Generate Clear Event Token

Functionality	**To get clear event Token**
Application	**To clear event**
Function name	getClearEventToken
Parameter list	

Parameter name	Type	Example (Length)	Description
meterNo	**String**	**String of 12 numbers**	**Meterno.**

	sgc	String	String of 6 numbers	Supply group code
	tariffIndex	int	1-99	Tariff index
	keyVersion	int	1-9	Key version
	keyExpiredTime	int	0-255	Key expire time
	keyNo	long	0-65535	Key Sequence
	seqNo	int	1-255	Token Sequence
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> Note: errorCode definition refer to chart 3. If succeeded, need return an array of <tokens>elements			
Function	public String getClearEventToken (String meterNo,			

definition	String Sgc, int tarrifIndex,
	int keyVersion, int keyExpiredTime, long keyNo, int seqNo)
	{
	...
	}

3.2.4.3 Generate Setup Max Power Limit Mode Token

Functionality	To get the set up Max Power Limit Mode Token			
Application	To set up Max Power Limit Mode Token			
Function name	getMaxPowerLimitToken			
Parameter list	Parameter name	Type	Scope and value	Description
	meterno.	String	String of 12 numbers	Meterno.
	sgc	String	String of 6 numbers	Supply group code
	tariffIndex	Int	1-99	Tariff index
	keyVersion	Int	1-9	Key version
	keyExpiredTime	Int	0-255	Key expire time
	keyNo	Long	0-65535	Key Sequence
	seq	Int	1-255	Token Sequence
	activatingModel	Int	0-1	Active

			0: Noramal Mode **1: Immediately switch mode**	**Mode** **This parameter always 1.**
	activatingDate	String	**YYYY-MM-DD**	**Active Date** Change Date type to String type: adapt to all kinds developmen t languages
	maxPowerLimit s	**int[]**	**1-2047 (array length :2) scale&unit:0.1kW**	**Max Power array**
	Hours	**Int[]**	**0-23 (array length :2)unit:hou r**	**Active time**
Return value	**Return XML String** **Xml format:** **<?xml version="1.0" encoding="UTF-8"?>** **<result>** **<errorCode></errorCode>** **<tokens>** **<token></token>** **...** **<token></token>** **<tokens>** **</result>**			

	Note: **errorCode definition refer to chart 3.** **If succeeded, need return an array of <tokens>elements**
Function definition	**public String getMaxPowerLimitToken (String meterNo, String Sgc,int tarrifIndex, int keyVersion, int keyExpiredTime , long keyNo, int seq , int activatingModel, String date int[] maxPowerLimits, int[] hours)** **{** **...** **}**

3.2.4.4 Generate Setup Single Tariff Token

Functionality	**To get the Setup Single Tarrif Token**			
Application	**To set up single tarrif**			
Function name	getSingleTariffToken			
Parameter list	**Name**	**Type**	**Scope or length**	**Description**
	meterNo	**String**	**String of 12 numbers**	**Meterno.**
	sgc	**String**	**String of 6 numbers**	**Supply group code**
	tariffIndex	**Int**	**1-99**	**Tariff index**
	keyVersion	**Int**	**1-9**	**Key version**

	keyExpiredTime	**Int**	**0-255**	**Key expire time**
	keyNo	**Long**	**0-65535**	**Key Sequence**
	seq	**Int**	**1-255**	**Token Sequence**
	activatingDate	String	**YYYY-MM-DD**	**Active date** Change Date type to String type: adapt to all kinds development languages
	activatingModel	**Int**	**0-1** **0: Normal Mode** **1: Immediately switch mode**	**Active Mode** **always 1.**
	validate	**Int**	**0-7**	**Validate always 0.**
	rate	**Int**	**1-8191(scale&unit: 0.01TK) By the integer bits and fractional bits, up to two decimal places, the maximum rate of price can be set to 81.91**	**Unit price**
Return	**Return XML String**			

value	**Xml format:** **<?xml version="1.0" encoding="UTF-8"?>** **<result>** **<errorCode></errorCode>** **<tokens>** **<token></token>** **...** **<token></token>** **<tokens>** **</result>** **Note:** **errorCode definition refer to chart 3.** **If succeeded, need return an array of <tokens>elements**
Function definition	**public String getSingleTariffToken(String meterNo, String Sgc, int tarrifIndex,** **int keyVersion, int keyExpiredTime , long keyNo, int seq,** **String activatingDate, int activatingModel,int validDate, int rate)** **{** **...** **}**

3.2.4.5 Generate Setup Step Tariff Token

Functionality	**To get the Setup Step Tarrif Token**
Application	**To set up step tarrif**
Function name	getStepTariffToken
Parameter	Parameters | Type | Scope&or length | Descriptio

list				
	meterNo	String	String of 12 numbers	Meterno.
	sgc	String	String of 6 numbers	Supply group code
	tariffIndex	int	1-99	Tariff index
	keyVersion	int	1-9	Key version
	keyExpiredTime	int	0-255	Key expire time
	keyNo	long	0-65535	Key Sequence
	seq	int	1-255	Token Sequence
	activatingDate	String	YYYY-MM-DD	Active date Change Date type to String type: adapt to all kinds development languages
	validate	int	0-7	Effective date
	activatingMod	Int	0-1	Effective

	el		0: Normal Mode 1: Immediately switch mode	Mode Always 1
	rates	int[]	Length 4 1-8191(scale&unit: 0.01 TK) By the integer bits and fractional bits, up to two decimal places, the maximum rate of price can be set to 81.91	Price array
	steps	int[]	1-4096 (the max array length :8) scale&unit:1 kWh	Step array
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> for example: 0-100 kWh rate: 2.3TK 100-200kWh rate: 2.7TK			

	200-300kWh rate: 3.2TK 300-∞ rate: 4.5TK rate[]= {2.3,2.7,3.2,4.5} steps[]={100,200,300} **If succeeded, need return an array of <tokens>elements**
Function definition	**public String getStepTariffToken(String meterNo, String Sgc, int tarrifIndex,int keyVersion, int keyExpiredTime, long keyNo, int seq,String activatingDate, int activatingModel, int validDate, int[] rates, int[] steps)** **{** **...** **}**

3.2.4.6 Generate Setup TOU Tariff Token

Functionality	**To get the Setup TOU tariff Token**
Application	**To set up TOU tariff**
Function name	getTOUTariffToken
Parameter list	

Parameter name	Type	Scope of field	Description
meterNo.	**String**	**String of 12 numbers**	**Meterno.**
Sgc	**String**	**String of 6 numbers**	**Supply group code**
tariffIndex	**Int**	**1-99**	**Tariff index**

keyVersion	**Int**	**1-9**	**Key version**
keyExpiredTime	**Int**	**0-255**	**Key expire time**
keyNo	**long**	**0-65535**	**Key Sequence**
Seq	**int**	**1-255**	**Token Sequence**
activatingDate	**String**	**YYYY-MM-DD**	**Active Date** Change Date type to String type: adapt to all kinds development languages
activatingModel	**Int**	**0-1** **0: Normal** **1: Immediately switch mode**	**Active Mode** **always 1.**
validate	**Int**	**0-7**	**Effective Date**
rates	**Int[]**	**Length 4** **1-8191(scale&unit: 0.01 TK)** **By the integer bits and**	**Price**

			fractional bits, up to two decimal places, the maximum rate of price can be set to 81.91	
	times	int[]	Length 2 0-23	Hours period
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> Note: errorCode definition refer to chart 3. If succeeded, need return an array of <tokens>elements			

Function definition	public String getTOUTariffToken(String meterNo, String Sgc, int tarrifIndex,int keyVersion, int keyExpiredTime, long keyNo, int seq,String activatingDate, int activatingModel, int validDate, int[] rates, int[] times)
	{
	...
	}			

3.2.4.7 Generate Setup friendly mode Token

Functionality	To get the Setup Friendly Mode Token
Application	To set up friendly mode
Function name	getFriendModeToken

Parameter list

Parameter name	Type	Scope or Length	Description
meterNo	String	String of 12 numbers	Meterno.
Sgc	String	String of 6 numbers	Supply group code
tariffIndex	int	1-99	Tariff index
keyVersion	int	1-9	Key version
keyExpiredTime	int	0-255	Key expire time
keyNo	long	0-65535	Key Sequence
seqNo	int	1-255	Token Sequence
friendMode	int	0-1 0: friendly mode is enable 1:friendly mode is closed	Friendly mode
Times	int[]	0-23	Friendly

			(array length :2)	**time interval**
	Days	**int[]**	**0-6 means Saturdy, Friday, Thursday , Wednesday, Tuesday , Monday and Sunday in sequence.** **(array length :7)**	**Weekly holiday** Array value 0 means weekend and 1 means ordinary day
	N_of_allowable_days	**int**	**0-99**	**No of allowable days that friendly hour will work after that without vending meter will not work in friendly hour**
Return value	**Return XML String** **Xml format:** **<?xml version="1.0" encoding="UTF-8"?>** **<result>** **<errorCode></errorCode>** **<tokens>** **<token></token>**			

	... <token></token> <tokens> </result> Note: errorCode definition refer to chart 3. If succeeded, need return an array of <tokens>elements

Function definition	public String getFriendModeToken(String meterNo, String Sgc, int tarrifIndex,
	int keyVersion, int keyExpiredTime , long keyNo, int seqNo, int friendMode, int[] times, int[] days)
	{
	...
	}

3.2.4.8 Generate Switch Meter Mode Token

Functionality	**To get Switch Meter Mode Token**			
Application	**To set up switch meter mode**			
Function name	getChangeMeterModeToken			
Parameter list	**Parameter name**	**Type**	**Scope or length**	**Description**
	meterNo	**String**	**String of 12 numbers**	**Meterno.**
	Sgc	**String**	**String of 6 numbers**	**Supply group code**

	tariffIndex	int	1-99	Tariff index
	keyVersion	int	1-9	Key version
	keyExpiredTime	int	0-255	Key expire time
	keyNo	long	0-65535	Key Sequence
	seqNo	int	1-255	Token Sequence
	mode	int	0-1 0:switch to the ordinary meter mode; 1:switch to the pre-paid meter mode	Meter mode switch
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result>			

	Note: **errorCode definition refer to chart 3.** **If succeeded, need return an array of <tokens>elements**
Function definition	**public String getChangeMeterModeToken(String meterNo, String Sgc,** **int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int mode)** **{** **...** **}**

3.2.4.9 Generate Set Credit Amount Limit And Overdraw Amount Limit Token

Functionality	**To get set credit amount limit and overdraw Amount Limit Token**
Application	**To set credit amount limit and overdraw Amount Limit .**
Function name	getSetCreditAmountLimitAndOverdrawAmountLimit

Parameter list

Parameter name	Type	Size/length	Description
meterNo	**String**	**String of 12 numbers**	**Meterno.**
Sgc	**String**	**String of 6 numbers**	**Supply group code**
tariffIndex	**Int**	**1-99**	**Tariff index**

	keyVersion	Int	1-9	Key version
	keyExpiredTime	Int	0-255	Key expire time
	keyNo	Long	0-65535	Key Sequence
	seqNo	Int	1-255	Token Sequence
	amountType	Int	0-1 0: creditAmountLimit 1: overAmountLimit	Balance Type
	amountLimit	Int	0~99999999(scale&unit:0.01TK)	Amount
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> Note:			

	errorCode definition refer to chart 3. **If succeeded, need return an array of <tokens>elements**
Function definition	**public String getSetCreditAmountLimitOrOverdrawAmountLimitToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, int seqNo, int amountType, int amountLimit)** **{** **...** **}**

3.2.4.10 Generate Logoff Token

Functionality	**To get reset Token**
Application	**To reset meter**
Function name	generateLogoffReturnToken
Parameter list	

Name	Type	Size/Length	Description
meterNo	**String**	**String of 12 numbers**	**Meterno.**
Sgc	**String**	**String of 6 numbers**	**Supply group code**
tariffIndex	**Int**	**1-99**	**Tariff index**
keyVersion	**Int**	**1-9**	**Key version**
keyExpiredTime	**Int**	**0-255**	**Key expire time**

	keyNo	Long	0-65535	Key Sequence
	seqNo	Int	1-255	Token Sequence
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> Note: errorCode definition refer to chart 3. If succeeded, need return an array of <tokens>elements			

Function definition	public String generateLogoffReturnToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, int keyNo, int seqNo)
	{
	...
	}			

3.2.4.11 Generate Credit/ Management Return Token

<table>
<tr><td>Functionality</td><td>To get return Credit/Management Return Token</td></tr>
<tr><td>Application</td><td>User should get the return token before credit Return ,and send the meter information to the electricity sale system, or else the electricity sale system will not sale electricity to user ,it will helpful for subsequent management 。</td></tr>
<tr><td>Function name</td><td>resolveReturnToke</td></tr>
<tr><td>Parameter list</td><td>
<table>
<tr><th>Parameter name</th><th>Type</th><th>String length</th><th>Description</th></tr>
<tr><td>meterNo</td><td>String</td><td>String of 12 numbers</td><td>Meterno.</td></tr>
<tr><td>Sgc</td><td>String</td><td>String of 6 numbers</td><td>SGCID</td></tr>
<tr><td>tariffIndex</td><td>int</td><td>1-99</td><td>Tariff index</td></tr>
<tr><td>keyVersion</td><td>int</td><td>1-9</td><td>Key version</td></tr>
<tr><td>keyExpiredTime</td><td>int</td><td>0-255</td><td>Key expire time</td></tr>
<tr><td>keyNo</td><td>long</td><td>0-65535</td><td>Key Sequence</td></tr>
<tr><td>Token</td><td>String</td><td>Custom defined by each manufacturers</td><td>Token encryption character String</td></tr>
</table>
</td></tr>
<tr><td>Return value</td><td>Return XML String
Xml format:
<code><?xml version='1.0' encoding='utf-8'?></code>
<code><result></code>
<code><type>0</type></code></td></tr>
</table>

```
<balance>454.54</balance>
<sequence>6</sequence>
<event>
<clockSetFlag>1</clockSetFlag>
<batteryVoltageLowFlag>0</batteryVoltageLowFlag>
<openCoverFlag>0</openCoverFlag>
<openBottomCoverFlag>1</openBottomCoverFlag>
<byPassFlag>0</byPassFlag>
<reverseFlag>0</reverseFlag>
<magneticInterfereFlag>0</magneticInterfereFlag>
<relayStatusFlag>1</relayStatusFlag>
<relayFaultFlag>0</relayFaultFlag>
<overdraftUsedFlag>0</overdraftUsedFlag>
<forwardActiveEnergyTol>11.23</forwardActiveEnergyTol>
<tariffIndex>51</tariffIndex>
</event>
</result>
```


Note:

If token resolve fail then return null.

Return returnTokens:

Parameter	Scope of Length	Description
Type	0-1 0:credit return token 1:logoff return token	
Balance		Balance of meter

Sequence	0-1 0:credit return token 1:logoff return token	Sequence of meter
clockSetFlag	0-1 0:unhappen 1:happen	Clock set event
batteryVoltageLowFlag	0-1 0:unhappen 1:happen	Battery low voltage event
openCoverFlag	0-1 0:unhappen 1:happen	Meter cover open event
openBottomCoverFlag	0-1 0:unhappen 1:happen	Terminal cover open event
byPassFlag	0-1 0:unhappen 1:happen	Bypass event
reverseFlag	0-1 0:unhappen 1:happen	Reverse event
magneticInterfereFlag	0-1 0:unhappen 1:happen	Magnetic interference event
relayStatusFlag	0-1	Relay status

		0:connect 1:disconnect	
	relayFaultFlag	0-1 0:unhappen 1:happen	Relay fault event
	overdraftUsedFlag	0-1 0:no 1:yes	Emergency balance used or not
Function definition	**public String resolveReturnToken (String meterNo, String Sgc, int tarrifIndex, int keyVersion, int keyExpiredTime, long keyNo, String Token)** **{** **...** **}**		

3.2.5 Test Function Interface Define

Functionality	To get test Token			
Application	To manage the user meter test and display, call the function and generate the test token.			
Function name	**getTestToken**			
Parameter list	Parameter name	Type	Scope or length	Description
	manufacturingID	Int	2 numbers	Manufacturer code Hexing:14
	control	Int	0 : test all the contents 1 : test relay 2 : test LCD display 3 : teat total energy 4 : test max power limit	Test token type

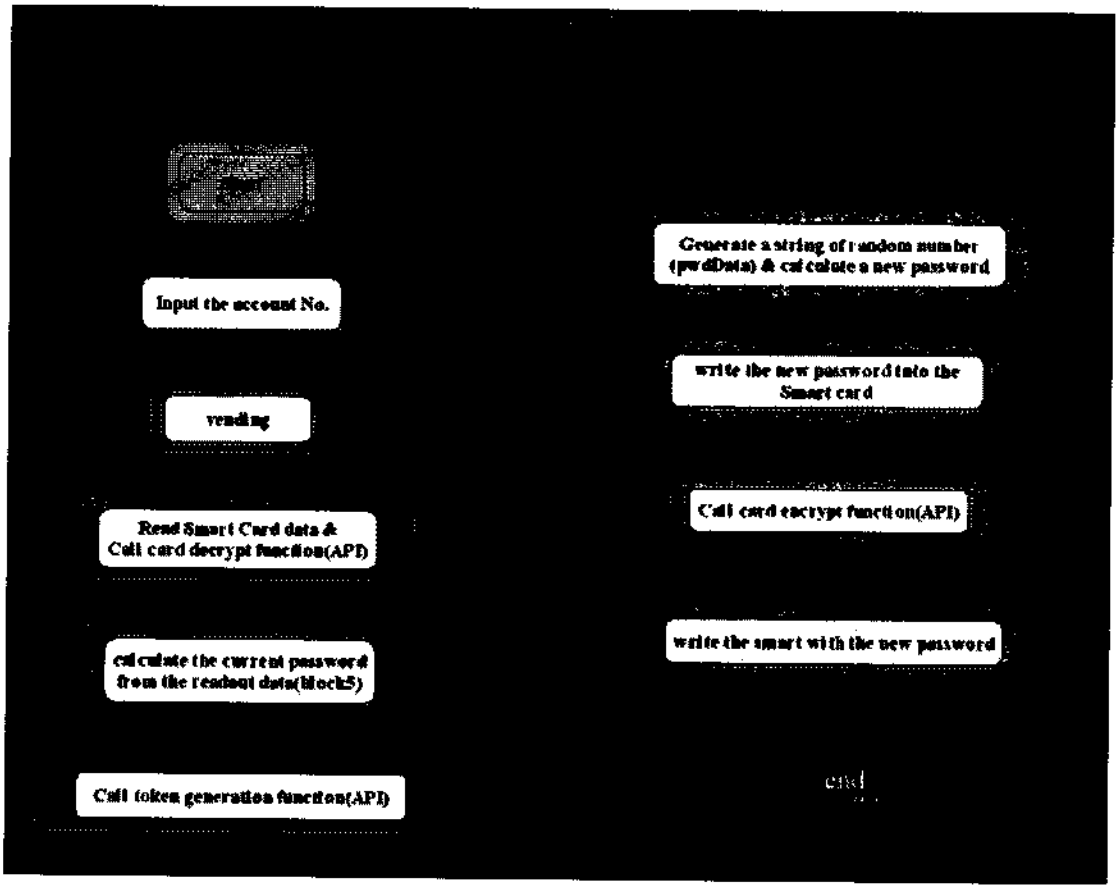
				5 : display current meter status 6 : display current power 7: display meter version number 8: display current tariff unit price 9: display overload current limit 10: display credit numbers 11: display serial number of meter 12: display off numbers of relay 13: enter into accuracy test mode 18-36: save	
Return value	Return XML String Xml format: <?xml version="1.0" encoding="UTF-8"?> <result> <errorCode></errorCode> <tokens> <token></token> ... <token></token> <tokens> </result> Note: errorCode definition refer to chart 3. If succeeded, need return an array of <tokens>elements				

Function definition	public String getTestToken (int manufacturingID, int control)
	{
	...
	}				

89

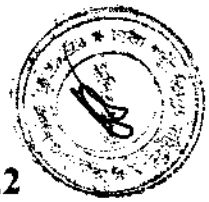
4 Functionality of Read And Write With Card

Smart Card business process



4.1 Smart Card Data Format

Data	***Bytes***	***Memory Address (Bytes)***
Non Encrypted Data		
Answer To Reset	4	0-4
Binary Pattern	2	4-6
Version	1	6-6
Meter ID	10	7-16
Consumer ID	10	17-26
Utility ID	6	27-32
Sanctioned Load	6	33-38
Meter Type	1	39-39
Sanctioned Load exceeded	2	40-41
Last Recharge Amount	4	42-45
Last Recharge Date	3	46-48
Last Transaction ID	10	49-58
Encrypted Data as Generated by the Meter Manufacturer SDK/API		
The following Data must exist within the encrypted Data: *Data To Meter:* Meter ID	940	100-1024



Consumer ID Utility ID Tokens Credit Token Management Token Test Token *Data From Meter:* Tamper Status RTC Status LR Status Usage Data for the last 6 months individually Month Year KWh Taka Recharged Taka Used Average Power Reactive Power Maximum Power Volt Ampere KWh in Peak KWh in Offpeak KVarh in Peak KVarh in Offpeak Total charge in Peak Total charge in Offpeak Number of power failures Number of time sanctioned load exceeded Tamper Number of times tampered Date/Time of Tamper Retrun Token		

4.2 Smart Card Standard Interface Define

Program language	JAVA
Functionality	Smart Card standard interface
Package	com.hx.ami.spi
Interface	SmartCard

Class definition	public interface SmartCard
	{
	// the encryption data for Smart card
	public String encryptSmartCardReturnData(int answerToReset, int binaryPattern, int version, String meterID, String customerID, String utilityID, long sanctionedLoad, int meterType, int lastRechargeAmount,

-88-

	String lastRechargeDate, String lastTransactionID, int cardType,String pwdData, int TokenTotalNumber, String[] tokens);
	
	note: the encryption data in the card address range:4-1019 size:1016bytes
	
	// Smart card to decrypt the return data
	public String decryptSmartCardReturnData(String meterNo, String sgc, int tariffIndex, int keyVersion, int keyExpiredTime, long keyNo, String encryptedData);
	}
	Note:The input parameter: encryptedData length is 1024 bytes,the whole card data

4.3 Encrypt Data Write to Smart Card Function Define

Functionality	Decrypt Smart Card return data
Application	To write the data to smart card , call function to encrypt function
Function name	**encryptSmartCardReturnData**
Parameter list	

[illegible]	[illegible]	[illegible]	[illegible]
answerToReset	Int	8	Answer To Reset Reserved position in IC card default set value 0
binaryPattern	String	Hex string String of 4 chars	Fixed string 'ACF0' set value ACF0
Version	Int	2	Version Manufacturer code Example: Hexing:14
meterNo	String	String of numbers Length<=13	Meterno.
customerID	String	String of 20 chars	Customer ID
utilityID	String	String of 4 chars	UtilityID
sanctionedLoad	long	0-65535	sanctionedLoad scale&unit:0.1kW
meterType	String	String of 2 numbers	Meter Type 01:keypad meter 02:card meter

				03:gprs meter
	lastRechargeAmount	Int	String of 4 numbers	Scale&Unit:0.01TK
	lastRechargeDate	String	String of 6 numbers	DDMMYY
	lastTransactionID	String	String of 20	
	cardType	String	String of 4 chars	OABC:user card 0BAC:test card
	pwdData	String	String of 12 chars	Put the first 12 chars pwdData into <block5>
	tokenTotalNumber	Int	String of 2 numbers	Max value25
	tokens	String[]	Every element length:String of 20 numbers or Greater than 20 numbers	
Return value	Return String **mapping in the card address range:4-1019 size:1016bytes 2032chars**			

Function definition	public String encryptSmartCardReturnData(int answerToReset, int binaryPattern, int version, String meterID, String customerID, String utilityID, long sanctionedLoad, int meterType, int lastRechargeAmount, String lastRechargeDate, String lastTransactionID, int cardType,String pwdData, int TokenTotalNumber, String[] tokens){
	...
	}			

4.4 Decrypt Return Data From Smart Card Function Define

Functionality	Decrypt Smart Card return data			
Application	To read the response data from smart card , call function to decrypt the datas.			
Function name	**decryptSmartCardReturnData**			
Parameter list	**Parameter name**	**Type**	**Suggested length**	**Description**
	meterNo	String	String of numbers Length<=13	Meter number

	Sgc	String	String of 6 numbers	SGCID	
	tariffIndex	Int	1-99	Tariff index	
	keyVersion	Int	1-9	Key version	
	keyExpiredTime	Int	0-255	Key expire time	
	keyNo	long	0-65535	Key Sequence	
	encryptedData	String	Custom manufacturers	Card String 1024bytes 2048char 4428 IC card length	

Return value

Return XML String

Xml format:

```
<?xml version="1.0" encoding="UTF-8"?>
<card>
    <block1>
        <answerToReset></answerToReset>
        <version></version>
        <meterID></meterID>
        <customerID></customerID>
        <utilityID></utilityID>
        <sanctionedLoad></sanctionedLoad>
        <meterType></meterType>
        <lastTransactionID></lastTransactionID>
        <lastRechargeDate></lastRechargeDate>
        <lastTransactionID></lastTransactionID>
    </block1>

    <block2>
        <cardType></cardType>
        <cardUsedFlag></cardUsedFlag>
        <tokenTotalNumber></tokenTotalNumber>
    </block2>

    <block3>
        <!—more than one token，split with ","-->
        <token></token>
    </block3>

    <block4>
        <meterID></meterID>
        <consumerID></consumerID>
        <utilityID></utilityID>
```

```
        </block4>

        <block5>
                <psData1></psData1>
                <psData2></psData2>
                <psData3></psData3>
                <psData4></psData4>
                <psData5></psData5>
                <psData6></psData6>
        </block5>

        <block6>
                <!—more than one token，split with “,”-->
                <tokenReturnCode></tokenReturnCode>
        </block6>

        <block7>
                <LastMonth1>
                        <billingDate> [redacted]</billingDate>
                        <activeEnergyImport></activeEnergyImport>
                        <takaRecharged></takaRecharged>
                        <takaUsed></takaUsed>
                        <activeMaximumPower></activeMaximumPower>
                        <reactiveMaximumPower></reactiveMaximumPower>
                        <activeEnergyImportT1></activeEnergyImportT1>
                        <activeEnergyImportT2></activeEnergyImportT2>
                        <reactiveEnergyImportT1></reactiveEnergyImportT1>
                        <reactiveEnergyImportT2></reactiveEnergyImportT2>
                        <totalChargeT1></totalChargeT1>
                        <totalChargeT2></totalChargeT2>
                        <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExc
eeded>

        <monthAveragePowerFactor></monthAveragePowerFactor>
                </LastMonth1>

                <LastMonth2>
                        <billingDate></billingDate>
                        <activeEnergyImport></activeEnergyImport>
                        <takaRecharged></takaRecharged>
                        <takaUsed></takaUsed>
                        <activeMaximumPower></activeMaximumPower>
```

```
                    <reactiveMaximumPower></reactiveMaximumPower>
                    <activeEnergyImportT1></activeEnergyImportT1>
                    <activeEnergyImportT2></activeEnergyImportT2>
                    <reactiveEnergyImportT1></reactiveEnergyImportT1>
                    <reactiveEnergyImportT2></reactiveEnergyImportT2>
                    <totalChargeT1></totalChargeT1>
                    <totalChargeT2></totalChargeT2>
                    <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExc
eeded>

        <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth2>

            <LastMonth3>
                <billingDate></billingDate>
                <activeEnergyImport></activeEnergyImport>
                <takaRecharged></takaRecharged>
                <takaUsed></takaUsed>
                <activeMaximumPower></activeMaximumPower>
                <reactiveMaximumPower></reactiveMaximumPower>
                <activeEnergyImportT1></activeEnergyImportT1>
                <activeEnergyImportT2></activeEnergyImportT2>
                <reactiveEnergyImportT1></reactiveEnergyImportT1>
                <reactiveEnergyImportT2></reactiveEnergyImportT2>
                <totalChargeT1></totalChargeT1>
                <totalChargeT2></totalChargeT2>
                <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExc
eeded>

        <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth3>

            <LastMonth4>
                <billingDate></billingDate>
                <activeEnergyImport></activeEnergyImport>
                <takaRecharged></takaRecharged>
                <takaUsed></takaUsed>
                <activeMaximumPower></activeMaximumPower>
                <reactiveMaximumPower></reactiveMaximumPower>
                <activeEnergyImportT1></activeEnergyImportT1>
```

```
                    <activeEnergyImportT2></activeEnergyImportT2>
                    <reactiveEnergyImportT1></reactiveEnergyImportT1>
                    <reactiveEnergyImportT2></reactiveEnergyImportT2>
                    <totalChargeT1></totalChargeT1>
                    <totalChargeT2></totalChargeT2>
                    <numberOfPowerFailures></numberOfPowerFailures>

      <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExc
eeded>

      <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth4>

            <LastMonth5>
                  <billingDate></billingDate>
                  <activeEnergyImport></activeEnergyImport>
                  <takaRecharged></takaRecharged>
                  <takaUsed></takaUsed>
                  <activeMaximumPower></activeMaximumPower>
                  <reactiveMaximumPower></reactiveMaximumPower>
                  <activeEnergyImportT1></activeEnergyImportT1>
                  <activeEnergyImportT2></activeEnergyImportT2>
                  <reactiveEnergyImportT1></reactiveEnergyImportT1>
                  <reactiveEnergyImportT2></reactiveEnergyImportT2>
                  <totalChargeT1></totalChargeT1>
                  <totalChargeT2></totalChargeT2>
                  <numberOfPowerFailures></numberOfPowerFailures>

      <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExc
eeded>

      <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth5>

            <LastMonth6>
                  <billingDate></billingDate>
                  <activeEnergyImport></activeEnergyImport>
                  <takaRecharged></takaRecharged>
                  <takaUsed></takaUsed>
                  <activeMaximumPower></activeMaximumPower>
                  <reactiveMaximumPower></reactiveMaximumPower>
                  <activeEnergyImportT1></activeEnergyImportT1>
                  <activeEnergyImportT2></activeEnergyImportT2>
                  <reactiveEnergyImportT1></reactiveEnergyImportT1>
```

```
                    <reactiveEnergyImportT2></reactiveEnergyImportT2>
                    <totalChargeT1></totalChargeT1>
                    <totalChargeT2></totalChargeT2>
                    <numberOfPowerFailures></numberOfPowerFailures>

      <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExc
eeded>

      <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth6>
      </block7>

      <block8>
            <returnToken></returnToken>
            <updateFlag></updateFlag>
      </block8>
</card>
```

Example with data

```
<?xml version='1.0' encoding='utf-8'?>
<card>
      <block1>
            <answerToReset>91102392</answerToReset>
            <version>51</version>
            <meterID>00000000037990020549</meterID>
            <customerID>00000000000000999910</customerID>
            <utilityID>0001</utilityID>
            <sanctionedLoad>000000000005</sanctionedLoad>
            <meterType>20</meterType>
            <LastRechargeAmount>412.22</LastRechargeAmount>
            <lastRechargeDate>150919</lastRechargeDate>

      <lastTransactionID>00000000000000012858</lastTransactionID>
      </block1>
      <block2>
            <cardType>0abc</cardType>
            <cardUsedFlag>fa</cardUsedFlag>
            <tokenTotalNumber>0</tokenTotalNumber>
      </block2>
      <block3>
            <token></token>
            <token></token>
            <token></token>
            <token></token>
            <token></token>
            <token></token>
            <token></token>
            <token></token>
            <token></token>
```

-4-


```
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
<token></token>
</block3>
<block4>

<meterID>0000000000000000000049050290790300000000</meterID>

<consumerID>0000000000000000000010999900000000000000</consumer
ID>
<utilityID>000000000100</utilityID>
</block4>
<block5>
<psData1>0c</psData1>
<psData2>71</psData2>
<psData3>53</psData3>
<psData4>12</psData4>
<psData5>f4</psData5>
<psData6>77</psData6>
</block5>
<block6>
<tokenReturnCode>0</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
<tokenReturnCode>255</tokenReturnCode>
```

```
                <tokenReturnCode>255</tokenReturnCode>
                <tokenReturnCode>255</tokenReturnCode>
                <tokenReturnCode>255</tokenReturnCode>
                <tokenReturnCode>255</tokenReturnCode>
                <tokenReturnCode>255</tokenReturnCode>
                <tokenReturnCode>255</tokenReturnCode>
        </block6>
        <block7>
                <LastMonth1>
                        <billingDate>20150920105212</billingDate>
                        <activeEnergyImport>0.0</activeEnergyImport>
                        <takaRecharged>0.0</takaRecharged>
                        <takaUsed>0.0</takaUsed>
                        <activeMaximumPower>0.0</activeMaximumPower>
                        <reactiveMaximumPower>0.0</reactiveMaximumPower>
                        <activeEnergyImportT1>0.0</activeEnergyImportT1>
                        <activeEnergyImportT2>0.0</activeEnergyImportT2>

        <reactiveEnergyImportT1>0.0</reactiveEnergyImportT1>

        <reactiveEnergyImportT2>0.0</reactiveEnergyImportT2>
                        <totalChargeT1>0.0</totalChargeT1>
                        <totalChargeT2>0.0</totalChargeT2>
                        <numberOfPowerFailures>1</numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded>0</numberOfSanctionedLoadExcee
ded>

        <monthAveragePowerFactor>0.0</monthAveragePowerFactor>
                </LastMonth1>
                <LastMonth2>
                        <billingDate></billingDate>
                        <activeEnergyImport></activeEnergyImport>
                        <takaRecharged></takaRecharged>
                        <takaUsed></takaUsed>
                        <activeMaximumPower></activeMaximumPower>
                        <reactiveMaximumPower></reactiveMaximumPower>
                        <activeEnergyImportT1></activeEnergyImportT1>
                        <activeEnergyImportT2></activeEnergyImportT2>
                        <reactiveEnergyImportT1></reactiveEnergyImportT1>
                        <reactiveEnergyImportT2></reactiveEnergyImportT2>
                        <totalChargeT1></totalChargeT1>
                        <totalChargeT2></totalChargeT2>
                        <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExceed
ed>

        <monthAveragePowerFactor></monthAveragePowerFactor>
                </LastMonth2>
                <LastMonth3>
                        <billingDate></billingDate>
                        <activeEnergyImport></activeEnergyImport>
```

```
                <takaRecharged></takaRecharged>
                <takaUsed></takaUsed>
                <activeMaximumPower></activeMaximumPower>
                <reactiveMaximumPower></reactiveMaximumPower>
                <activeEnergyImportT1></activeEnergyImportT1>
                <activeEnergyImportT2></activeEnergyImportT2>
                <reactiveEnergyImportT1></reactiveEnergyImportT1>
                <reactiveEnergyImportT2></reactiveEnergyImportT2>
                <totalChargeT1></totalChargeT1>
                <totalChargeT2></totalChargeT2>
                <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExceed
ed>
        <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth3>
            <LastMonth4>
                <billingDate></billingDate>
                <activeEnergyImport></activeEnergyImport>
                <takaRecharged></takaRecharged>
                <takaUsed></takaUsed>
                <activeMaximumPower></activeMaximumPower>
                <reactiveMaximumPower></reactiveMaximumPower>
                <activeEnergyImportT1></activeEnergyImportT1>
                <activeEnergyImportT2></activeEnergyImportT2>
                <reactiveEnergyImportT1></reactiveEnergyImportT1>
                <reactiveEnergyImportT2></reactiveEnergyImportT2>
                <totalChargeT1></totalChargeT1>
                <totalChargeT2></totalChargeT2>
                <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExceed
ed>
        <monthAveragePowerFactor></monthAveragePowerFactor>
            </LastMonth4>
            <LastMonth5>
                <billingDate></billingDate>
                <activeEnergyImport></activeEnergyImport>
                <takaRecharged></takaRecharged>
                <takaUsed></takaUsed>
                <activeMaximumPower></activeMaximumPower>
                <reactiveMaximumPower></reactiveMaximumPower>
                <activeEnergyImportT1></activeEnergyImportT1>
                <activeEnergyImportT2></activeEnergyImportT2>
                <reactiveEnergyImportT1></reactiveEnergyImportT1>
                <reactiveEnergyImportT2></reactiveEnergyImportT2>
                <totalChargeT1></totalChargeT1>
                <totalChargeT2></totalChargeT2>
                <numberOfPowerFailures></numberOfPowerFailures>

        <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExceed
```

	ed> <monthAveragePowerFactor></monthAveragePowerFactor> </LastMonth5> <LastMonth6> <billingDate></billingDate> <activeEnergyImport></activeEnergyImport> <takaRecharged></takaRecharged> <takaUsed></takaUsed> <activeMaximumPower></activeMaximumPower> <reactiveMaximumPower></reactiveMaximumPower> <activeEnergyImportT1></activeEnergyImportT1> <activeEnergyImportT2></activeEnergyImportT2> <reactiveEnergyImportT1></reactiveEnergyImportT1> <reactiveEnergyImportT2></reactiveEnergyImportT2> <totalChargeT1></totalChargeT1> <totalChargeT2></totalChargeT2> <numberOfPowerFailures></numberOfPowerFailures> <numberOfSanctionedLoadExceeded></numberOfSanctionedLoadExceed ed> <monthAveragePowerFactor></monthAveragePowerFactor> </LastMonth6> </block7> <block8> <returnToken>07485085822316193422</returnToken> <updateFlag>fa</updateFlag> </block8> </card> Note: Determine the element< errorCode> is greater than or equal to 0, if the conditions established, generate Token error. Otherwise, the build succeeds, you can call get an array of <tokens>elements.

Function definition	public String decryptSmartCardReturnData(String meterNo, String sgc, int tariffIndex, int keyVersion, int keyExpiredTime, long keyNo, String encryptedData)
	{
	...
	}

4.5 Smart Card Writing Password Algorithm

Below is the example of password block:

```
<block5>
            <psData1>0c</psData1>
            <psData2>71</psData2>
            <psData3>53</psData3>
            <psData4>12</psData4>
            <psData5>f4</psData5>
```

```
        <psData6>77</psData6>
    </block5>
```

Hex total = psData1* 255 + psData2* 100 + psData3* 155 + psData4* 10 + psData5* 55 + psData6;

Then the password will be last two bytes of the total; One exception is if all the data is equal to FF then password will be FFFF. `<psData1>0c</psData1>` here 0c is Hex format.

4.6 The format of byte after decryption (Just for example):

Data	Length	Address	Data Type	Is Encrypted	Desc
Answer To Reset	4	0			Answer To Reset
Binary Pattern	2	4			Fixed Data 0b10101100111100 00 = 0xACF0
Version	1	6	BCD	N	Manufacturers
Meter ID	10	7	BCD	N	Meterno.
Consumer ID	10	17	BCD	N	A/C No.
Utility ID	2	27	BCD	N	Utility ID
Sanctioned Load	6	29	BCD	N	Sanctioned Load
Meter Type	1	35	HEX	N	Meter Type
Sanctioned Load exceeded	1	36	HEX	N	Sanctioned Load exceeded
Last Recharge Amount	2	37	HEX	N	Last Recharge Amount
Last Recharge Date	3	39	BCD	N	
Last Transaction ID	10	42	BCD	N	Last Transaction ID
Card Type	2	100	HEX	N	0ABC-Account Card, 0BAC-Test Card B, 0AAC-Test Card A
Card Used Flag	1	102	Boolean	N	Token in the card has been processed flag. Upon completion of Token processing, write the results, and then update the flag FA. POS point writing Token set the

					flag FF
Token Total Number	1	103	HEX	N	Token Total Numbe
[redacted]				N	Data sum check
	6	106		N	Blank Area:FF
Token 1	10	112	BCD	N	Token area, arranged in strict accordance with the required number in order of priority. No Token fill the whole area F
Token 2	10	122	BCD	N	
Token 3	10	132	BCD	N	
Token 4	10	142	BCD	N	
Token 5	10	152	BCD	N	
Token 6	10	162	BCD	N	
Token 7	10	172	BCD	N	
Token 8	10	182	BCD	N	
Token 9	10	192	BCD	N	
Token 10	10	202	BCD	N	
Token 11	10	212	BCD	N	
Token 12	10	222	BCD	N	
Token 13	10	232	BCD	N	
Token 14	10	242	BCD	N	
Token 15	10	252	BCD	N	
Token 16	10	262	BCD	N	
Token 17	10	272	BCD	N	
Token 18	10	282	BCD	N	
Token 19	10	292	BCD	N	
Token 20	10	302	BCD	N	
Token 21	10	312	BCD	N	
Token 22	10	322	BCD	N	
Token 23	10	332	BCD	N	
Token 24	10	342	BCD	N	
Token 25	10	352	BCD	N	
	6	362		N	

Meter ID	20	368	BCD	N	Meter ID
Consumer ID	20	388	BCD	N	Consumer ID
Utility ID	6	408	BCD	N	Utility ID
				N	
	0	416		N	
Data A	1	416	HEX	N	
Data B	1	417	HEX	N	
Data C	1	418	HEX	N	
Data D	1	419	HEX	N	
Data E	1	420	HEX	N	
Data F	1	421	HEX	N	
				N	
	0	424		N	
Token 1 Return Code	1	424	HEX	Y	Meter operating results after Token, 0x00: success, 0xFF: default value, the other: the error code.
Token 2 Return Code	1	425	HEX	Y	
Token 3 Return Code	1	426	HEX	Y	
Token 4 Return Code	1	427	HEX	Y	
Token 5 Return Code	1	428	HEX	Y	
Token 6 Return Code	1	429	HEX	Y	
Token 7 Return Code	1	430	HEX	Y	
Token 8 Return Code	1	431	HEX	Y	
Token 9 Return Code	1	432	HEX	Y	
Token 10 Return Code	1	433	HEX	Y	
Token 11 Return	1	434	HEX	Y	

Code					
Token 12 Return Code	1	435	HEX	Y	
Token 13 Return Code	1	436	HEX	Y	
Token 14 Return Code	1	437	HEX	Y	
Token 15 Return Code	1	438	HEX	Y	
Token 16 Return Code	1	439	HEX	Y	
Token 17 Return Code	1	440	HEX	Y	
Token 18 Return Code	1	441	HEX	Y	
Token 19 Return Code	1	442	HEX	Y	
Token 20 Return Code	1	443	HEX	Y	
Token 21 Return Code	1	444	HEX	Y	
Token 22 Return Code	1	445	HEX	Y	
Token 23 Return Code	1	446	HEX	Y	
Token 24 Return Code	1	447	HEX	Y	
Token 25 Return Code	1	448	HEX	Y	
Recharge Date&Time	6	449	HEX	Y	SSMMHHDDMMYY, Meter recharge time, prepayment System writes all F
Recharge Amount	4	455	HEX	Y	unit:0.01, The amount of recharge
[redacted]				Y	

-22-

	3	461	HEX	Y	
Billing Date&Time	6	464	BCD	Y	SSMMHHDDMMYY
Active energy import (Current Month)	4	470	BCD	Y	unit:0.01 kWh
Taka Recharged(Current Month)	4	474	HEX	Y	Unit 0.01
Taka Used(Current Month)	4	478	HEX	Y	unit 0.01
Active Maximum Power (MD)	3	482	BCD	Y	Active Maximum Power (MD)
Reactive Maximum Power (MD)	3	485	BCD	Y	Reactive Maximum Power (MD)
Active energy import T1(Current Month)	4	488	BCD	Y	Active energy import T1(Current Month)
Active energy import T2(Current Month)	4	492	BCD	Y	Active energy import T2(Current Month)
Reactive energy import T1(Current Month)	4	496	BCD	Y	Reactive energy import T1(Current Month)
Reactive energy import T2(Current Month)	4	500	BCD	Y	Reactive energy import T2(Current Month)
Total Charge T1(Current Month)	4	504	HEX	Y	Unit:0.01
Total Charge T2(Current Month)	4	508	HEX	Y	unit:0.01

Number of Power Failures	2	512	HEX	Y	Number of Power Failures
Number of Sanctioned Load Exceeded	2	514	HEX	Y	Number of Sanctioned Load Exceeded
Month Average Power Factor	2	516	BCD	Y	
[redacted]				Y	
	0	520		Y	
Billing Date&Time	6	520	BCD	Y	SSMMHHDDMMYY
Active energy import (Current Month)	4	526	BCD	Y	unit:0.01 kWh
Taka Recharged(Current Month)	4	530	HEX	Y	Unit:0.01
Taka Used(Current Month)	4	534	HEX	Y	Unit:0.01
Active Maximum Power (MD)	3	538	BCD	Y	
Reactive Maximum Power (MD)	3	541	BCD	Y	
Active energy import T1(Current Month)	4	544	BCD	Y	
Active energy import T2(Current Month)	4	548	BCD	Y	
Reactive energy import T1(Current Month)	4	552	BCD	Y	
Reactive energy import	4	556	BCD	Y	

-29-

T2(Current Month)					
Total Charge T1(Current Month)	4	560	HEX	Y	Unit:0.01
Total Charge T2(Current Month)	4	564	HEX	Y	Unit:0.01
Number of Power Failures	2	568	HEX	Y	
Number of Sanctioned Load Exceeded	2	570	HEX	Y	
Month Average Power Factor	2	572	BCD	Y	
[redacted]				Y	
	0	576		Y	
Billing Date&Time	6	576	BCD	Y	SSMMHHDDMMYY
Active energy import (Current Month)	4	582	BCD	Y	unit:0.01 kWh
Taka Recharged(Current Month)	4	586	HEX	Y	Unit:0.01
Taka Used(Current Month)	4	590	HEX	Y	Unit:0.01
Active Maximum Power (MD)	3	594	BCD	Y	
Reactive Maximum Power (MD)	3	597	BCD	Y	
Active energy import T1(Current Month)	4	600	BCD	Y	
Active energy import	4	604	BCD	Y	

T2(Current Month)					
Reactive energy import T1(Current Month)	4	608	BCD	Y	
Reactive energy import T2(Current Month)	4	612	BCD	Y	
Total Charge T1(Current Month)	4	616	HEX	Y	Unit:0.01
Total Charge T2(Current Month)	4	620	HEX	Y	
Number of Power Failures	2	624	HEX	Y	
Number of Sanctioned Load Exceeded	2	626	HEX	Y	
Month Average Power Factor	2	628	BCD	Y	
[redacted]				Y	
	0	632		Y	
Billing Date&Time	6	632	BCD	Y	SSMMHHDDMMYY
Active energy import (Current Month)	4	638	BCD	Y	unit:0.01 kWh
Taka Recharged(Current Month)	4	642	HEX	Y	Unit:0.01
Taka Used(Current Month)	4	646	HEX	Y	Unit:0.01
Active Maximum Power (MD)	3	650	BCD	Y	
Reactive	3	653	BCD	Y	

Maximum Power (MD)					
Active energy import T1(Current Month)	4	656	BCD	Y	
Active energy import T2(Current Month)	4	660	BCD	Y	
Reactive energy import T1(Current Month)	4	664	BCD	Y	
Reactive energy import T2(Current Month)	4	668	BCD	Y	
Total Charge T1(Current Month)	4	672	HEX	Y	Unit:0.01
Total Charge T2(Current Month)	4	676	HEX	Y	Unit:0.01
Number of Power Failures	2	680	HEX	Y	
Number of Sanctioned Load Exceeded	2	682	HEX	Y	
Month Average Power Factor	2	684	BCD	Y	
[redacted]				Y	
	0	688		Y	
Billing Date&Time	6	688	BCD	Y	SSMMHHDDMMYY
Active energy import (Current Month)	4	694	BCD	Y	unit:0.01 kWh
Taka	4	698	HEX	Y	Unit:0.01

Recharged(Current Month)					
Taka Used(Current Month)	4	702	HEX	Y	Unit:0.01
Active Maximum Power (MD)	3	706	BCD	Y	
Reactive Maximum Power (MD)	3	709	BCD	Y	
Active energy import T1(Current Month)	4	712	BCD	Y	
Active energy import T2(Current Month)	4	716	BCD	Y	
Reactive energy import T1(Current Month)	4	720	BCD	Y	
Reactive energy import T2(Current Month)	4	724	BCD	Y	
Total Charge T1(Current Month)	4	728	HEX	Y	Unit:0.01
Total Charge T2(Current Month)	4	732	HEX	Y	Unit:0.01
Number of Power Failures	2	736	HEX	Y	
Number of Sanctioned Load Exceeded	2	738	HEX	Y	
Month Average Power Factor	2	740	BCD	Y	

				Y	
	0	744		Y	
Billing Date&Time	6	744	BCD	Y	SSMMHHDDMMYY
Active energy import (Current Month)	4	750	BCD	Y	unit:0.01 kWh
Taka Recharged(Current Month)	4	754	HEX	Y	Unit:0.01
Taka Used(Current Month)	4	758	HEX	Y	Unit:0.01
Active Maximum Power (MD)	3	762	BCD	Y	
Reactive Maximum Power (MD)	3	765	BCD	Y	
Active energy import T1(Current Month)	4	768	BCD	Y	
Active energy import T2(Current Month)	4	772	BCD	Y	
Reactive energy import T1(Current Month)	4	776	BCD	Y	
Reactive energy import T2(Current Month)	4	780	BCD	Y	
Total Charge T1(Current Month)	4	784	HEX	Y	Unit:0.01
Total Charge T2(Current	4	788	HEX	Y	Unit:0.01

Month)					
Number of Power Failures	2	792	HEX	Y	
Number of Sanctioned Load Exceeded	2	794	HEX	Y	
Month Average Power Factor	2	796	BCD	Y	
[redacted]				Y	
	0	800		Y	
Numbers of open cover	2	800	HEX	Y	
The last time of open cover	6	802	BCD	Y	SSMMHHDDMMYY
The last 2 time of open cover	6	808	BCD	Y	
The last 3 time of open cover	6	814	BCD	Y	
The last 4 time of open cover	6	820	BCD	Y	
[redacted]				Y	
	4	828		Y	
Numbers of open terminal cover	2	832	HEX	Y	
The last time of open terminal cover	6	834	BCD	Y	
The last 2 time of open terminal cover	6	840	BCD	Y	
The last 3 time of open terminal cover	6	846	BCD	Y	
The last 4 time of open terminal cover	6	852	BCD	Y	
[redacted]				Y	
	4	860		Y	
Numbers of bypass	2	864	HEX	Y	
The last time of	6	866	BCD	Y	

bypass					
The last 2 time of bypass	6	872	BCD	Y	
The last 3 time of bypass	6	878	BCD	Y	
The last 4 time of bypass	6	884	BCD	Y	
[redacted]				Y	
	4	892		Y	
Numbers of overload	2	896	HEX	Y	
The last time of overload	6	898	BCD	Y	
The last 2 time of overload	6	904	BCD	Y	
The last 3 time of overload	6	910	BCD	Y	
The last 4 time of overload	6	916	BCD	Y	
[redacted]				Y	
	4	924		Y	
Numbers of missing neutral event	2	928	HEX	Y	
The last time of missing neutral event	6	930	BCD	Y	
The last 2 time of missing neutral event	6	936	BCD	Y	
The last 3 time of missing neutral event	6	942	BCD	Y	
The last 4 time of missing neutral event	6	948	BCD	Y	
[redacted]				Y	
	4	956		Y	
Last Read Time	6	960	BCD	N	POS needs to reset the data

					after read the date, set all F
Return Token	10	966	BCD	N	POS needs to reset the tokens, set all F
Update Flag	1	976	Boolean	N	The meter is updated to return data, FF-no, FA-updated. POS card is required to write the flag is after read the data
[redacted]				N	
Verify Writable	1	1020	HEX		PSC used to check the password is corrected
Error Counter	1	1021			Card Reserved fields
PSC	2	1022			Card Reserved fields

5 GPRS Meters Connection Platform Design

5.1Concept Description of MIC Communication Access

Smart GPRS meters communicate with MIC using XML format based on TCP. It will use encryption and digital signature technical in TCP linker layer, and in data area of XML document, follow the standard DLMS protocol.

XML Communication Protocol Define

XML Format Define

```
<? xml version='1.0' ?>
<h:rt xmlns h='ProtocolHead'>
<h:pv>1</h:pv>
    <h:addr>030140000170</h:addr>
    <h:dir>down</h:dir>
    <h:pt>1</h:pt>
```

<h:fc>3</h:fc>
<h:seq>7</h:seq>
<h:e>10</h:e>
<h:a>0</h:a>
<h:r>485845110000000000000007</h:r>< h:d >
0A15CEDF16.....0A15CEDF16
</h:d>
<h:sg>0203<h:sg>
</h:rt>

XML Label Description:

All data of XML should be transferred using ASCII code.

Protocol Presentation	Protocol Description	Note
<? xml version='1.0' ?>	XMLfixed head format	
<h:rt xmlns h='ProtocolHead'>	To name the space under root element.Protocol Head means the first layer of protocol.	
<h:pv>1</ h:pv>	XMLprotocol internal version number of company. 1: the first version	
< h:f>100000000</h:f>	Address of transmit end, the label means which transmit end the data come from. The address will be 1 if the transmit end is master station; The address will be meter address if the transmit end is meter, whose length is not fixed and the max length is 32 pcs ASCII code.	
<h:dir>down</h:dir>	The direction of transmission. up: meter to server down: server to meter	
<h:pt>1</h:pt >	Data transport protocol in data area. 1: DLMS transport protocol 2: XML transport protocol (Company will develop in the future, as the second stage)	
<h:fc>1</h:fc>	Function code: it is mainly for front-end processor distinguishes the data types. When front-end processor doesn't analysis the data or decrypt the encrypted data, it can response the frame transmitted from the meters. At the same time, the label can distinguish who is the transmit end. 1: Link interface test(meter launches) 2: Link interface test response frame(master station responses) 3: Request – response frame(master station responses) 4: Request —confirm/deny frame (master station responses) 5: Report initially- confirm/deny frame (meter launches) 6: Report initially confirm frame needed (meter launches) 7: Report initially confirm frame needless(meter launches) 8: business password refresh (master station launches) 9: manufacturer request (mainly for upgrade remotely) (master station launches) 10: manufacturer respond (mainly for upgrade remotely)) (master station launches)	
< h:seq>3< h:seq>	XML frame serial number, for defining which response frame was responded by the response end. The frame serial number is changed by primary end, it will add 1 when primary end launches one frame of message. The range of variationcan be0-99. Driven end regards the frame serial number of	

	primary end as the response frame serial number. After messages transmitted by starting end, when the responded can't be received timely , the serial number of retransmission will be same if the starting end allows retransmission, and the max retransmission numbers is 3; 2) If the driven end receives two starting frame with the same serial number in series, it means the response hasn't been received yet, then retransmission(and without dealing with the message again); 3) If starting end receives two response frames with the same serial number in series, then will not deal with the second response frame.	
< h:e>10</h:e>	Data encryption algorithm 00: without using encryption algorithm , data transmitted using plaintext form 10: AES-128 In GCM Mode 11: AES-192 12: AES-256 20: DES 30: SHA 40: ECC	
< h:a>1</h:a>	Verify algorithm 0: without using verify 1: SHA-256	
< h:r>4827762</h:r>	Random number. The starting end will regard the random number as vector to encrypt the data area, the receiving end will regard the random number as vector to decrypt the data area. It has 12 bytes, as 24 bytes HEX character String in XML.	
< h:d >123456< /h:d >	Data area, if without using encryption algorithm, the flag is plaintext ; if using encryption algorithm, the flag is the plaintext will be encrypted.	
<h:sg>0203<h:sg>	When without using verify, the flag is useless; when using verify, the flag is verify code for plaintext of data area.	
</h:rt>	XML end flag.	

5.2 Function Code Description:

Link interface test: the function is distinguished in data area.

01: log in

02: quit

03: heartbeat

Link interface test response frame: the function is distinguished in data area.

01: log in

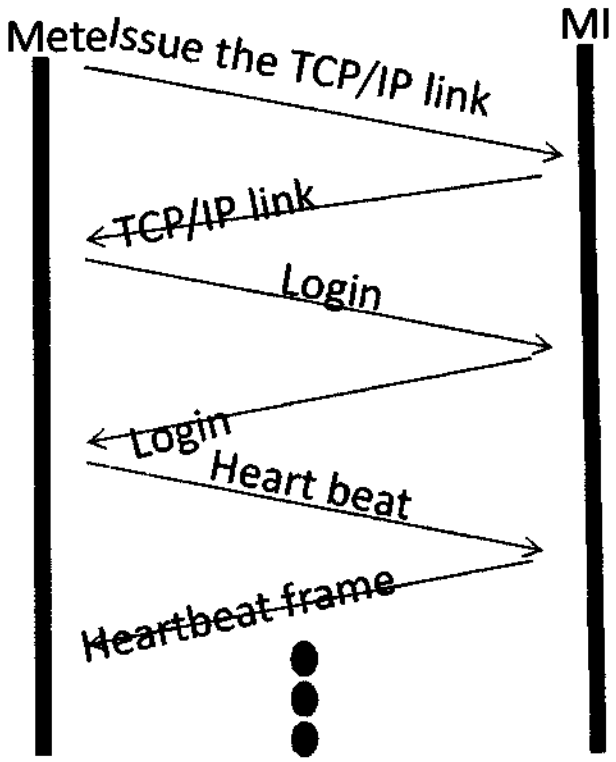
02: quit

03: heartbeat

In this system, meters as client side, MIC as server side, link build , log in request and heatbeat are transmitted by meters , and responded by MIC.

Log in frame: For transmitting a group of specific data to the MIC and confirming the link chaining relationship with master station , after meters connected with master station in TCP/IP.

Heartbeat frame: For transmitting a group of data to maintain the TCP/IP link chaining .



Request – response frame

Request – response frame is the request frame transmitted from master station and the response frame responded by meters.

The correct response means in XML layers, encryption and decryption runs correct or other data to be tested are correct, without including part or all deny for DLMS frames in DLMS layers.

Request- confirm\deny frame

Request- confirm\deny frame is the deny frame responded to the master station when errors happened on data that meters transmitted to the master station.

It will response confirm frame if the update key request to master station is correct.

Request- confirm\deny frame format:

Confirm frame	Function code:00	
Deny frame	Function code :01	Error code: 01: password error; 02: verify error;03: other error

Report initially-confirm\deny frame

For data meters transmitted to the master station , if need confirmation, front-end processor will using the function code to achieve confirmation and deny for meters.

The format refers to function code 3.

Report initially -confirm frame needed

For data meters reported to the master station initially, if need confirmed , sent according to the function code.

Report initially -confirm frame needless

For data meters reported to the master station initially, if confirmed needless , sent according to the function code.

Business password updated

To update the business password according to the function , sent by master station , encrypted by root key.

Manufacturer request

For self defined commands send from master station to the manufacturers.

Manufacturer response

For self defined commands responded from meters to the manufacturers.

It adopts plaintext transmission mode for data areas of function code 1,2,4,5.

Further information please check< The process of online meter communication >