# مفهوم پیام (message) در سیستم مدیریت فرایندها

پیام (message) یک مجموعه داده ساختاریافته است که تحت یک عنوان از یک فرستنده به یک گیرنده مشخص ارسال می شود. این داده ها، در گیرنده دریافت و پردازش می شود.

در سیستم مدیریت فرایندها، فرستنده عموما از جنس کار است. یعنی اینکه پیامها اغلب از طریق کارها ارسال می شود اما ارسال پیام محدود و منحصر به کار نیست. برای مثال از طریق رابط کاربری هم می توانیم پیامها را ارسال کنیم. گیرنده ها هم می توانند از جنس کار یا گردش کار باشند. یعنی اینکه پیام می تواند به یک یا چند کار خاص ارسال شود یا اینکه در سطح گردش کار باشد و به کل کارهای یک گردش کار ارسال شود. حال که با مفهوم کلی پیام آشنا شدیم بهتر است با هدف و کارکرد پیام آشنا شویم

## هدف و کارکرد پیام

بطور کلی یکی از اهداف اصلی پیام، تغییر حالت در گیرنده است. تغییر حالت مفهوم بسیار عامی است که شامل تمام ویژگیهای یک گردش کار است. از وضعیت (status) فعلی گرفته تا مقادیر و محتوای فیلدها، کاربران مجازی و حقیقی، حالت انتظار یا ادامه کار، اولویت کار، وضعیت تعلیق و نیز رویدادهایی مانند ارسال اعلامیه و ... را در بر می­­‌گیرد. بنابراین یک پیام می تواند در گردش کار گیرنده، هرگونه فعالیت تعریف شده و معتبر در سیستم مدیریت فرایندها را انجام دهد. البته یک پیام ممکن است به تغییر حالت نیانجامد و برای مثال فقط کارکرد اطلاع رسانی داشته باشد یا مثلا ممکن است پیام دریافت شود اما بنا به هردلیل هیچ تغییری در گیرنده انجام نشود.

**نکته مهم:** پیامها تنها برای کارهای فعال معنی دارند و برای کارهای غیرفعال (لغوشده، خاتمه یافته) نمی توان پیام ارسال کرد.

پس از آشنایی با مفهوم و کارکرد کلی پیامها، در ادامه با روند کلی پیامها از ارسال تا دریافت و پردازش آن آشنا می­شویم.

## روند ارسال و دریافت پیام

مراحل روند کلی پیام از زمان ارسال تا دریافت به شکل زیر است:

1. ارسال پیام به شکل دستی (رابط کاربری) یا از طریق برنامه و با استفاده از تابع WorkSendMessage
2. قرار گرفتن پیام در وضعیت در انتظار بررسی
3. تحویل پیام به گیرنده
4. عملیاتی شدن پیام در گیرنده از طریق تابع WorkProcessMessage
5. دریافت توسط وضعیت دریافت پیام

**توضیح:** بسته به پیام، ممکن است یکی از مراحل 4 یا 5 اتفاق بیافتد. البته در مواردی ممکن است هر دو مرحله هم وجود داشته باشد.

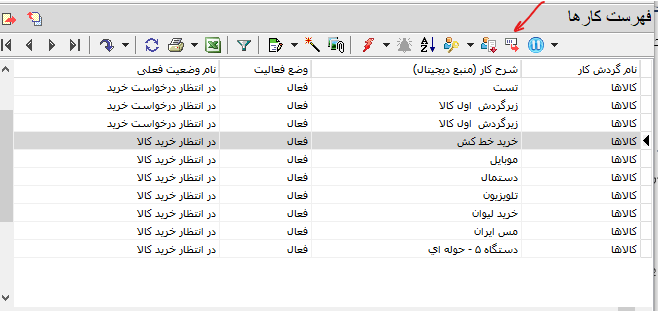
حال برای آشنایی دقیق تر، هر کدام از مراحل با جزئیات کامل شرح داده می شود:

### مرحله اول: ارسال پیام

همانطور که قبلا ذکر شد، ارسال پیام به دو روش دستی و از طریق برنامه انجام می شود که هر کدام از این روشها مزایا و ویژگیهای خاص خود را دارد. در ادامه به توضیح این دو روش می پردازیم:

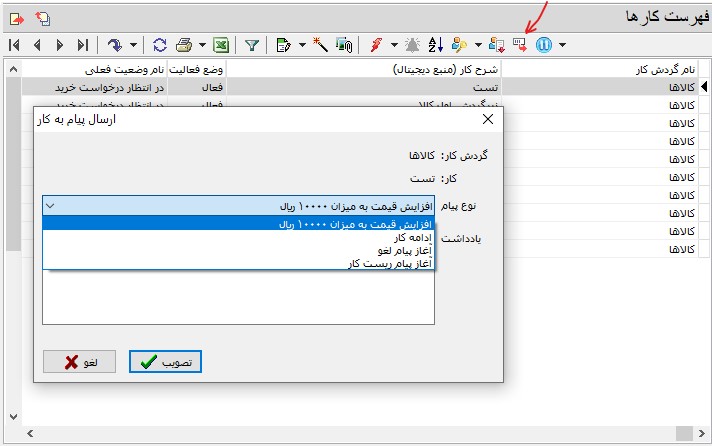
#### از طریق رابط کاربری و به شکل دستی

برای این کار، در محاوره فهرست کارها می توانید روی کار مورد نظر خود کلیک کنید و سپس در نوار ابزار روی آیکون ارسال پیام  کلیک کنید (شکل زیر):

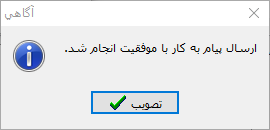


در این مثال، نشانگر روی کار "خرید خط کش" است و به این ترتیب قرار است که پیام به این کار خاص ارسال شود. حال با کلیک روی آیکون ارسال پیام که در شکل بالا با فلش مشخص شده است کلیک کنیم آنگاه محاوره انتخاب پیام (ارسال پیام به کار) به شکل زیر روی صفحه باز می شود. در این شکل، فهرست کشویی "نوع پیام" باز شده و لیست پیامهای قابل ارسال به روش دستی را نمایش می دهد:

**توضیح :** دقت کنید که برای این کار ابتدا باید فهرست پیامهای دستی را به گردش کار معرفی کرده باشید. نحوه معرفی پیامهای دستی به گردش کار در ضمیمه 1 با جزئیات کامل توضیح داده شده است.



حال می توانید پیام مورد نظر خود را انتخاب و سپس، تکمه تصویب را کلیک کنید. در ادامه محاوره ای باز می شود که تایید مجدد کاربر را می خواهد و با تایید این محاوره، پیام ارسال می شود و در انتها پیغام زیر ظاهر می شود:



نمایش این پیغام به این معنی است که پیام در وضعیت انتظار قرار گرفته است.

**نکته 1:** هر گردش کار، فهرست پیامهای دستی خاص خود را خواهد داشت که با گردش کارهای دیگر متفاوت خواهد بود.

**نکته 2:** با توجه به آنچه که در نکته 1 گفته شد، محدودیت اصلی در روش ارسال پیام به شکل دستی، محدود بودن این روش به گردش کار است. یعنی این که از این طریق در هر گردش کار، فقط می توانیم پیامهای دستی تعریف شده در همان گردش کار را ارسال کنیم. در ادامه مطلب و زمانی که با عدم این محدودیت در روش ارسال از طریق برنامه آشنا شدید، این محدودیت برای شما روشن تر خواهد شد.

#### ارسال پیام از طریق برنامه

کار مبدا (فرستنده)، با استفاده از تابع WorkSendMessage یک پیام به شکل بسته اطلاعاتی شامل کلید حرفی و پارامترهای حرفی و عددی، به کار مقصد(گیرنده) می فرستد. دقت کنید که شناسه اصلی و unique پیام، عبارت کلیدی پیام است و گیرنده ها پیامهای مختلف را از طریق این کلید، شناسایی و پردازش می کنند. برای درک بهتر ساختار پیام به الگوی تابع WorkSendMessage توجه کنید:

**function WorkSendMessage (dbName, data, numberParam, stringParam)**

**dbName**: همانطور که می بینید پارامتر اول این تابع، dbName یا نام پایگاه داده (سیستم اطلاعاتی) است و به این معنی است که از نظر تئوری امکان ارسال پیام بین پایگاهی هم وجود دارد. توضیح این که اگر این مقدار تعیین نشود به معنی پایگاه جاری است.

**data**: این پارامتر یک فایل xml با مشخصات زیر است:

<\_RQST WorkKey="xxx" MessageKeyStr="xxx" Note="xxx">

</\_RQST>

مشاهده می کنید که در این قسمت، مقصد دقیق پیام ارسالی با کلید کار مقصد (WorkKey) مشخص می­شود، ضمنا عبارت کلیدی پیام (MessageKeyStr) و در صورت لزوم، یک یادداشت دلخواه (Note) برای پیام تعیین می گردد.

**numberParam**: در اینجا پارامتر عددی پیام وارد می شود. اگر اطلاعات ارسالی فقط یک عدد باشد آنگاه می توانیم برای این کار از پارامتر عددی استفاده کنیم. کاربرد دیگر پارامتر عددی برای تفکیک پیامهای یکسان است. برای مثال فرض کنید یک پیام با عبارت کلیدی و پارامتر حرفی مشابه قرار است در شرایط مختلف دو رفتار متفاوت داشته باشد. در این حالت از پارامتر عددی برای تفکیک این دو رفتار استفاده می شود.

**stringParam**: در اینجا پارامتر حرفی پیام وارد می شود. پارامتر حرفی اطلاعاتی است که در گیرنده، پردازش می شود و براساس این اطلاعات، رفتار یا فعالیت مورد نظر انجام می شود.

**نکته:** در نظر داشته باشید که ارسال پیام نیاز به دسترسی از طریق برنامه به گردش کار مقصد دارد، بنابراین اگر در گردش کار مقصد (گیرنده) برای امکان ارسال پیام از طریق برنامه، نام کاربری و کلمه عبور تعیین شده باشد، در ابتدای کد تابع اجرایی برای ارسال پیام و قبل از تابع WorkSendMessage حتما نام کاربری و کلمه عبور معتبر برای WorkUtils را تعیین و وارد کنید. در غیر این صورت پیام، ارسال نخواهد شد و با پیغام نادرست بودن نام کاربری یا کلمه عبور مواجه خواهید شد. برای نمونه به مثال زیر توجه کنید:

function WorkTaskCaptureUser() {

WorkUtils.UserName=”tari”;

WorkUtils.Password="1234";

var data=’ **<\_RQST WorkKey="510" MessageKeyStr="capture" Note=""></\_RQST>**’;

WorkUtils.WorkSendMessage(“”, data, 0, “”);

}

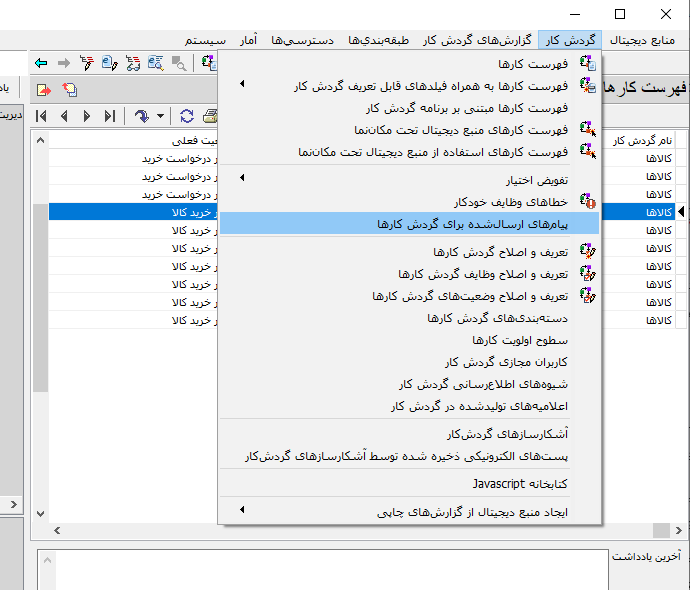
**توضیح:**  مفهوم پارامترهای حرفی و عددی در ابتدا و در مرحله تنوری شاید کمی مبهم به نظر بیاید. به همین خاطر، این مفاهیم در بخشهای بعدی و با ذکر مثالهای واقعی، بیشتر توضیح داده شده است.

### مزایای ارسال پیام از طریق برنامه

* **عدم محدودیت به پایگاه و گردش کار:** ارسال پیام از طریق برنامه محدود به گردش کار نیست. یعنی این که می توانیم از یک گردش کار به یک گردش کار دیگر پیام ارسال کنیم. علاوه بر آن محدودیت پایگاه (سیستم اطلاعاتی) هم ندارد یعنی از گردش کارهای یک پایگاه می توانیم به گردش کارهای یک پایگاه دیگر پیام ارسال کنیم.
* **امکان خودکار شدن ارسال پیام:** همانند فعالیتهای دیگر، ارسال پیام را نیز می توانیم در یک وظیفه خودکار (System Task) پیاده سازی کنیم تا در شرایط مورد نظر بطور اتوماتیک و از طریق سیستم ارسال شود.
* **امکان ارسال گروهی پیام:** اگر بخواهیم یک پیام خاص را به گروهی از کارها ارسال کنیم این کار فقط از طریق برنامه نویسی امکان پذیر است.

## مرحله دوم: قرار گرفتن پیام در انتظار بررسی

پیام پس از ارسال، در وضعیت انتظار قرار می گیرد. برای ملاحظه فهرست پیامهای ارسال شده، کافی است منوی "گردش کار" را باز کنید و از آنجا روی گزینه "پیامهای ارسال شده برای گردش کارها" کلیک کنید (شکل زیر):



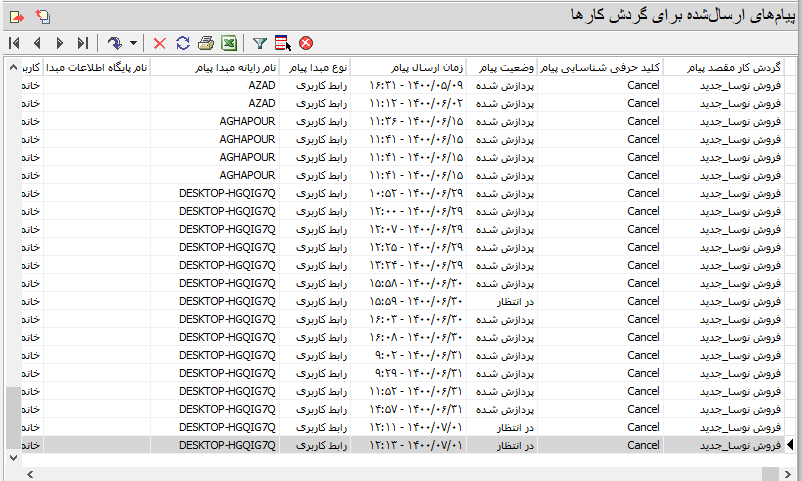
به این ترتیب فهرست پیامهای ارسال شده به کارها به شکل زیر نمایان می شود:



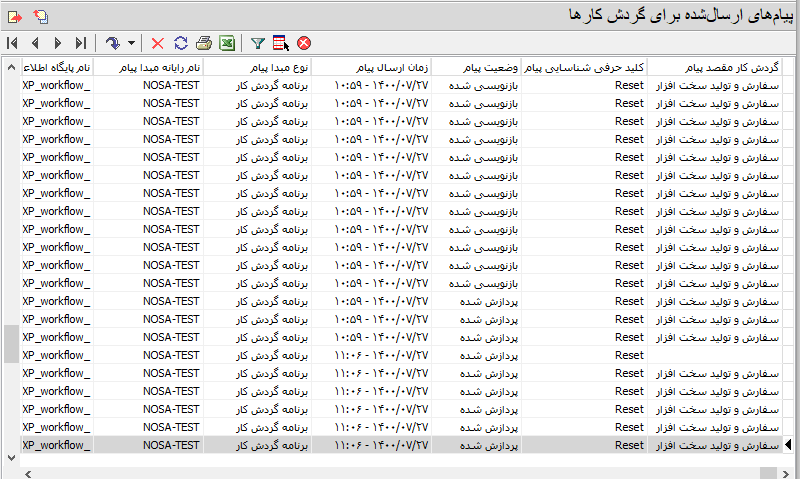
ملاحظه می کنید که در این فهرست، پیام ها در وضعیت "در انتظار" قرار دارند.

## مرحله سوم: تحویل پیام به گیرنده

در این مرحله پیام به کار(های) مقصد (گیرنده) تحویل می شود. در این حالت، وضعیت پیام در فهرست پیامهای ارسال شده، از وضعیت **در انتظار** به وضعیت **پردازش شده** تغییر می یابد. در شکل زیر پیامهای تحویل شده (پردازش شده) را ملاحظه می کنید:



**نکته مهم:** اگر یک پیام مشابه را همزمان چند بار و به صورت تکراری ارسال کنید، تنها آخرین ارسال، پردازش می شود و ارسالهای قبلی در وضعیت **بازنویسی شده** قرار می گیرند. برای نمونه به شکل زیر توجه کنید:



## مرحله چهارم: عملیاتی شدن پیام در گیرنده

این عملیات در تابع WorkProcessMessage و برمبنای داده های پیام (عبارت کلیدی پیام و پارامترهای عددی و حرفی پیام) انجام می شود. به عبارت بهتر، هر برنامه ای که در این تابع توشته شده است انجام می شود. حال سوال این است که داده های پیام چگونه به تابع WorkProcessMessage پاس می شود؟

جواب این است که داده های اصلی پیام (کلید حرفی پیام، پارامتر عددی و پارامتر حرفی) به صورت مشخصات یک موجود بنام WorkMessage در اختیار کار مقصد (گیرنده) قرار می گیرد. این موجود دارای سه ویژگی (Property) به شکل زیر است:

WorkMessage = {

KeyStr String/RO عبارت کلیدی پیام

NumberParam Number/RO پارامتر عددی

StringParam String/RO پارامتر حرفی

}

بنابراین برای استفاده از هرکدام از ویژگیهای پیام در کد نویسی، باید به شکل زیر عمل کنید:

عبارت کلیدی پیام : WorkMessage.KeyStr

پارامتر عددی: WorkMessage.NumberParam

پارامتر حرفی: WorkMessage.StringParam

**نکته 1:** در هر گیرنده، یک و تنها یک تابع WorkProcessMessage وجود دارد و تمام پیامها باید در این تابع پردازش شود. برای پردازش پیامهای مختلف، می توانیم در این تابع، با شرط گذاری روی عبارت کلیدی پیام (WorkMessage.KeyStr)، عملیات پردازش هر پیام را تفکیک کنیم.

**نکته 2:** همانطور که قبلا هم گفته شد، هیچ محدودیتی روی نوع فعالیت و اثرگذاری پیام در گیرنده وجود ندارد. یعنی این که هر نوع فعالیت (یا هر اسکریپتی) که در سیستم مدیریت فرایندها قابل تعریف و معتبر باشد، می تواند در تابع WorkProcessMessage پیاده سازی و عملیاتی شود. انواع این فعالیتها می تواند شامل تغییر وضعیت کار، تغییر محتوا و مقادیر فیلدهای کار، ارسال اعلامیه پیامک یا ایمیل، تعلیق/ رفع تعلیق کار، تعیین اولویت کار، دستور ادامه کار، لغو کار، خاتمه کار و ...باشد.

## مرحله 5: دریافت پیام توسط وضعیت آغاز پیام

در این مرحله، مسیر عادی فرایند متوقف می شود و کار در هر وضعیتی که هست به وضعیت آغاز پیام منتقل می شود. و سپس از این وضعیت به یک وضعیت دلخواه منتقل می شود. برای مثال ممکن است به وضعیتهای قبلی بازگردد یا چند وضعیت را رد کرده و به وضعیت دلخواه منتقل شود. این مرحله در صفحات بعد و در بخش **تغییر وضعیت (state) کار توسط پیام** با ذکر مثال و با جزئیات کامل شرح داده شده است.

## تقسیم بندی پیامها از جنبه پارامتر

پیامها از این نظر به دو گروه تقسیم می شود: پیامهای بدون پارامتر و پیامهای حاوی پارامتر.

**پیامهای بدون پارامتر:** اگر تکلیف پیام در گیرنده مشخص باشد اصطلاحا می گوییم این پیام بدون پارامتر است یعنی این که برای گیرنده فقط عبارت کلیدی پیام مهم است و به اطلاعات دیگری نیاز ندارد.

**پیامهای پارامتریک:** اگر برای پردازش پیام، علاوه بر عبارت کلیدی، به اطلاعات دیگری که توسط فرستنده ارسال می شود، نیاز داشته باشیم، آنگاه می گوییم این دسته از پیامها پارامتریک هستند. یعنی اینکه تکلیف پیام در گیرنده به خودی خود مشخص نیست و با پارامترهای تکمیلی که فرستنده در لحظه ارسال می کند تعیین می شود.

حال بهتر است با ذکر مثالهای واقعی از انواع پیامها، با جزئیات نحوه ارسال، دریافت و پردازش این پیامها بیشتر آشنا شویم:

## ارسال پیامها بدون پارامتر

این نوع از پیامها به دلیل معین بودن مقادیر، نیازی به پارامتر ندارند. نحوه ارسال این دسته از پیامها به شکل دستی و از طریق رابط کاربری قبلا اشاره شد، حال وقت آن است که نحوه ارسال پیامهای غیرپارامتریک از طریق برنامه را توضیح دهیم:

#### نحوه ارسال پیامهای غیرپارامتریک از طریق برنامه

همانطور که اطلاع دارید، ارسال پیامها در برنامه، از طریق تابع WorkSendMessage انجام می شود. حال برای یادآوری به ساختار کلی این تابع توجه کنید:

WorkSendMessage (dbName, data, numberParam, stringParam)

با توجه به این که قرار است پیام بدون پارامتر باشد لذا این تابع بدون پارامتر عددی و پارامتر حرفی خواهد بود. یعنی به این شکل:

WorkSendMessage (“”, data);

بنابراین یک تابع اجرایی نمونه برای ارسال این پیام می تواند به شکل زیر باشد:

function WorkTaskCaptureUser() {

WorkUtils.UserName=”tari”;

WorkUtils.Password="1234";

var data=’ **<\_RQST WorkKey="510" MessageKeyStr="capture" Note=""></\_RQST>**’;

WorkUtils.WorkSendMessage(“”, data, 0, “”);

}

یعنی این پیام، با عبارت کلیدی "capture" و بدون پارامتر، به کار با کلید 510 ارسال می شود.

### پردازش پیامهای غیرپارامتریک در گیرنده (WorkProcessMessage)

طبق آنچه که قبلا گفته شد، نحوه عملیات و پردازش پیام در گیرنده، در تابع WorkProcessMessage پیاده سازی می شود. حال به پیاده سازی دو مثال غیرپارامتریک در این تابع می پردازیم:

* **مثال 1، قاپیدن کار**

منظور از قاپیدن کار این است که یک ذینفع (کاربر) بتواند با ارسال یک پیام، کار بخصوصی را از کارتابل یک کاربر دیگر، به کارتابل خود منتقل کند. این مورد در فرایندهایی است که در یک وضعیت خاص، چند کاربر می توانند یک وظیفه خاص را انجام دهند و اینکه کدام کاربر کار را انجام دهد مهم نیست.

برای مثال در یک بخش فروش، چند کارشناس هستند که تایید پیش فاکتور از طرف مشتری را پیگیری و ثبت می کنند. اگرچه برای مدیریت بهتر کارها میان کارشناسان تقسیم می شود اما واقعا خیلی مهم نیست که کدام کارشناس، تایید پیش فاکتور را ثبت کند. در این وضعیت اگر مشتری تماس بگیرد و کارشناس مرتبط با کار او به هردلیل حاضر نباشد، همکار او می تواند کار را به کارتابل خودش منتقل کند و تایید پیش فاکتور را در سیستم ثبت کند.

حال ببینیم این مورد در تابع WorkProcessMessage چگونه پیاده سازی شده است:

function WorkProcessMessage() {

var key= WorkMessage.KeyStr;

if (key ==”capture”) {

WorkUserUtils.AddVirtualMemberList(“Nosa\\WorkPerformer”, Workstate1.WorkKey, WorkUserUtils.CurrentUser)

}

}

به شرطی که در این برنامه وجود دارد دقت کنید (if (key=”capture”))، این شرط تعیین می کند که اگر یک پیام با عبارت کلیدی "capture" به این کار فرستاده شود، آنگاه کاربر جاری (CurrentUser) به فهرست کاربران Nosa\\WorkPerformer افزوده می شود. توضیح اینکه WorkPerformer نام کاربر مجازی است که اختیار انجام وظیفه در کار را دارد. به این ترتیب کاربر جاری نیز اختیار انجام وظیفه را خواهد داشت و کار را در کارتابل خود مشاهده خواهد کرد.

* **مثال 2، افزودن قیمت یک کالا به میزان معین (مثلا 10000 ریال)**

برای این کار یک پیام با عبارت کلیدی (Add10000) را ارسال می کنیم. برای پردازش این پیام، باید یک شرط به تابع WorkProcessMessage اضافه کنید:

function WorkProcessMessage() {

var key=WorkMessage.KeyStr;

if (key ==”capture”) {

WorkUserUtils.AddVirtualMemberList(“Nosa\\WorkPerformer”, Workstate1.WorkKey, WorkUserUtils.CurrentUser)

} else if (key==”Add10000”) {

WorkState1.kalaPrice+=10000;

}

}

شرط و کدهای پردازش این پیام با رنگ آبی مشخص شده است. در این ناحیه تعیین می شود که اگر عبارت کلیدی پیام (Add10000) باشد آنگاه 10000 ریال به مقدار فیلد قیمت کالا (kalaPrice) افزوده شود.

## ارسال پیامهای پارامتریک

به دلیل محدودیتهای رابط کاربری، امکان ارسال پیامهای پارامتریک به شکل دستی (از طریق رابط کاربری) وجود ندارد و برای این کار باید از امکان عملیات خاص استفاده کنید. برای آشنایی با عملیات خاص به مطلب آموزشی **طراحی و کاربرد عملیات خاص در گردش کار** مراجعه کنید. به این ترتیب به شرح ارسال پیامهای پارامتریک از طریق برنامه می پردازیم:

### ارسال پیامهای پارامتریک از طریق برنامه

مثال 1- افزایش قیمت کالا به میزان ثابت (3000 ریال)

به این مثال دقت کنید

function WorkTaskPriceUp() {

WorkUtils.UserName=”tari”

WorkUtils.Password="1234";

var data=’<\_RQST WorkKey="510" MessageKeyStr="Price2" Note=""></\_RQST>’;

WorkUtils.WorkSendMessage(“”, data, 0, “3000”);

}

در این پیام، پارامتر حرفی، یک رشته حرفی است با محتوای "3000"و این دقیقا همان مبلغی است که می خواهیم به قیمت کالا اضافه کنیم . حال ببینیم نحوه پردازش این پیام را در تابع WorkProcessMessage چگونه است:

function WorkProcessMessage() {

var key=WorkMessage.KeyStr;

if (key ==”capture”) {

WorkUserUtils.AddVirtualMemberList(“Nosa\\WorkPerformer”, Workstate1.WorkKey, WorkUserUtils.CurrentUser)

} else if (key==”Add10000”) {

WorkState1.kalaPrice+=10000;

} else if (key==”Price2”) {

var ad=WorkMessage.numberParam;

WorkState1.kalaPrice+=ad;

}

}

قسمتی که با رنگ آبی مشخص شده، کدی است که برای این پیام اضافه شده است. در این کد، مقدار پارامتر عددی استخراج و به فیلد قیمت کالا (kalaPrice) افزوده می شود.

**توضیح:** در اینجا چون داده ارسالی فقط یک عدد صحیح است بنابراین از پارامتر عددی برای ارسال اطلاعات استفاده شد. پس بر این اساس به یک قاعده کلی می رسیم.

**قاعده کلی:** هر گاه از طریق پیام بخواهیم تنها یک عدد صحیح را به عنوان پارامتر ارسال کنیم، آنگاه بهتر است برای این کار از پارامتر عددی استفاده کنیم.

**مثال برای پارامتر حرفی متغیر**

حال برای روشن تر شدن موضوع، این بار پیام را با پارامتر حرفی متغیر ارسال می کنیم. برای مثال فرض کنید که مقدار افزایش قیمت برابر با حاصلضرب میزان تورم در قیمت کالا باشد. در این حالت، میزان تورم (که متغیر است) با پیام ارسال می شود و بر این اساس، قیمت افزایش می یابد. در این صورت پیام به شکل زیر ارسال می شود.

function WorkTaskPriceUp() {

WorkUtils.UserName=”tari”;

WorkUtils.Password="1234";

var data=’<\_RQST WorkKey="510" MessageKeyStr="Price2" Note=""></\_RQST>’;

var inflationIndex=WorkState2.inflInd;

WorkUtils.WorkSendMessage(“”, data, 0, inflationIndex);

}

و به این ترتیب، تابع WorkProcessMessage نیز به شکل زیر خواهد بود:

function WorkProcessMessage() {

var key=WorkMessage.KeyStr;

if (key ==”capture”) {

WorkUserUtils.AddVirtualMemberList(“Nosa\\WorkPerformer”, Workstate1.WorkKey, WorkUserUtils.CurrentUser)

} else if (key==”Add10000”) {

WorkState1.kalaPrice+=10000;

} else if (key==”Price2”) {

var p=WorkMessage.StringParam;

var ad= (parsInt(p) \* kalaPrice); // میزان افزایش قیمت برابر است با حاصل ضرب نرخ تورم در قیمت کالا

WorkState1.kalaPrice+=ad;

}

}

### ارسال پیامها با پارامتر حرفی پیچیده

پارامتر حرفی پیامها تا اینجا و در مثالهایی که زده شد، حاوی یک مقدار (ثابت یا متغیر) بود. حال فرض کنید بخواهیم تعداد بیش از یک مقدار را با پارامتر حرفی ارسال کنیم. در واقع تعدد و تنوع مقادیر موجود در پارامتر حرفی، امکان عملیات در گیرنده را توسعه می دهد. در این موارد پارامتر حرفی به شکل یک Object ارسال می شود که شامل چند Property (مقدار) می باشد. به یک مثال در این زمینه توجه کنید:

### مثال 2: افزایش همزمان تعداد و قیمت کالا

در این مثال، پارامتر حرفی باید شامل دو مقدار باشد: میزان افزایش کالا، و مبلغ افزایش قیمت. تابع ارسال این پیام با عبارت کلیدی KalaEdit به شکل زیر خواهد بود:

function WorkTaskUpdateKala() {

WorkUtils.UserName=”tari”;

WorkUtils.Password="1234";

var data=’ **<\_RQST WorkKey="510" MessageKeyStr="KalaEdit" Note=""></\_RQST>**’;

var msg={};

msg.addNum=20;

msg.addPrice=25000;

WorkUtils.WorkSendMessage(“”, data, 0, JSON.stringify(msg));

}

در این تابع، ابتدا یک متغیر از نوع Object و با نام msg تعریف می شود و سپس دو Property برای افزایش تعداد (addNum) و افزایش قیمت (addPrice) برای این Object تعریف می شود. در انتها msg با استفاده از متد stringify از JSON به یک رشته حرفی تبدیل شده و به گیرنده ارسال می شود. توضیح این که پارامترهای حرفی دارای محتوای مرکب و چندگانه باید حتما با این فرمت ارسال شود.

حال قرار است این پیام به گیرنده ارسال شود و در اثر آن، تعداد کالا 20 واحد و قیمت آن 25000 واحد افزایش یابد. به این ترتیب، کد این پیام به شکل زیر باید به تابع WorkProcessMessage اضافه شود:

function WorkProcessMessage() {

var key=WorkMessage.KeyStr;

if (key ==”capture”) {

WorkUserUtils.AddVirtualMemberList(“Nosa\\WorkPerformer”, Workstate1.WorkKey, WorkUserUtils.CurrentUser)

} else if (key==”Add10000”) {

WorkState1.kalaPrice+=10000;

} else if (key==”Price2”) {

var p=WorkMessage.StringParam;

var ad= (p \* kalaPrice); // میزان افزایش قیمت برابر است با حاصل ضرب نرخ تورم در قیمت کالا

WorkState1.kalaPrice+=ad;

} else if (key==”KalaEdit”) {

p=JSON.parse(WorkMessage.StringParam);

WorkState1.numbers+=p.addNum;

WorkState1.kalaPrice+=p.addPrice;

}

توضیح این که ابتدا باید با متد JSON.parse پارامتر حرفی به یک Object تبدیل می شود و سپس Property های این Object به فیلدهای تعداد (numbers) و قیمت کالا (addPrice) افزوده می شود.

**توضیح:** در این مثال برای سادگی، Propertyهای پارامتر حرفی (مقادیر افزایش تعداد و قیمت کالا)، مقادیر ثابت انتخاب شده است اما همانند مثال قبل، در این مثال هم می توانیم Property ها را به شکل متغیر تعریف کنیم.

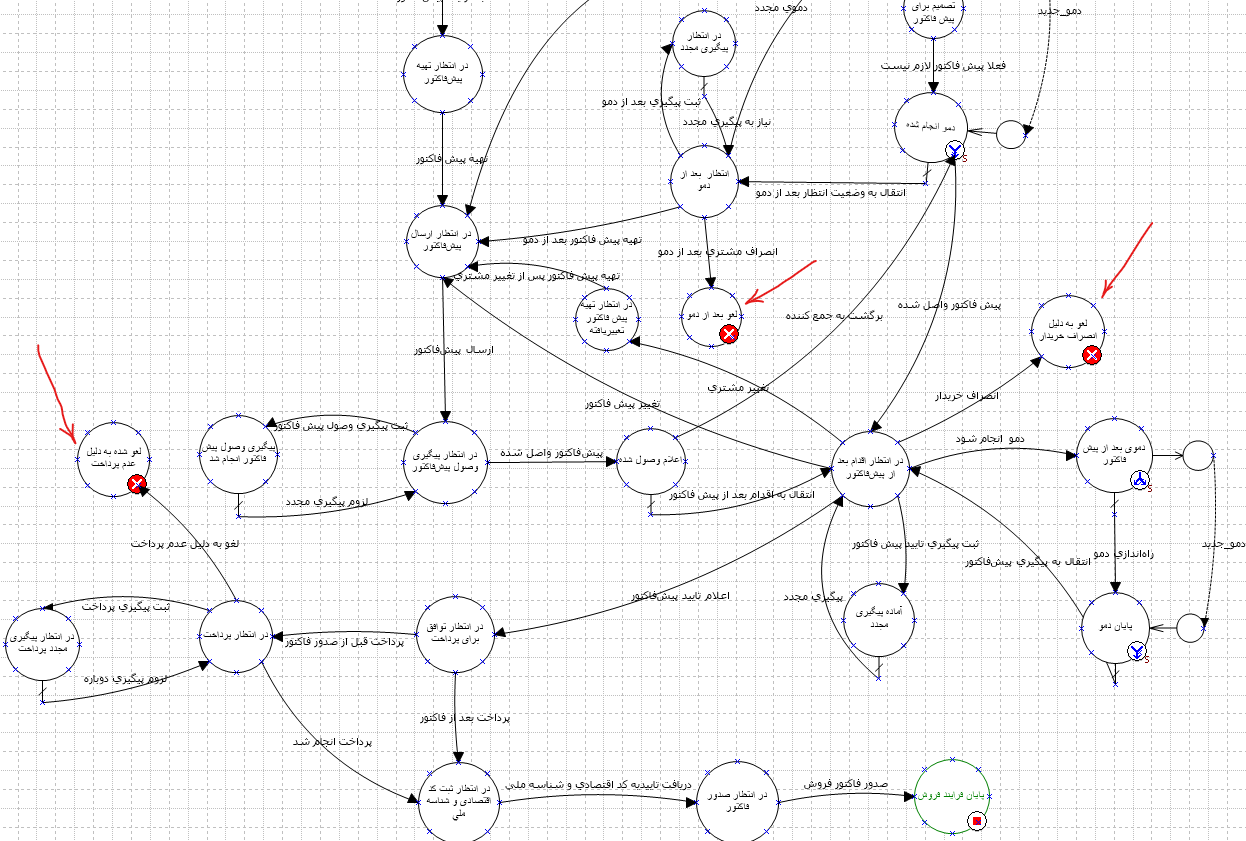
حال که با پاره ای از کارکردهای پیام آشنا شدیم بهتر است با یکی از کارکردهای دیگر پیام که تغییر وضعیت کار می باشد آشنا شویم

## تغییر وضعیت (State) کار از طریق پیام

یکی از کارکردهای پیام این است که می تواند مسیر و روند عادی فرایند را تغییر داده و بلافاصله کار را به یک وضعیت دلخواه دیگر منتقل کند. برای مثال می تواند کار مقصد را در هر وضعیتی که هست، به وضعیت خاتمه یا لغو منتقل کند.

**مثال: لغو کار از طریق پیام**

در تحلیل فرایند گردش کار، معمولا شرایط لغو کار بصورت سیستماتیک پیش بینی و پیاده سازی می شود. برای مثال به دیاگرام گردش کار فروش توجه کنید:



همانطور که مشاهده می کنید، در سه وضعیت پیش شده است که کار لغو می شود:

* مشتری پس از این که در یک جلسه دمو شرکت می کند، اعلام می کند که از خرید نرم افزار منصرف شده است.
* مشتری پیش فاکتور را تایید نمی کند و به این وسیله انصراف مشتری اعلام می شود.
* مشتری وجه یا پیش پرداخت را واریز نمی کند و به این ترتیب کار لغو می شود.

حال ممکن است در بعضی از وضعیتها، شرایط پیش بینی نشده ای پیش بیاید که نیاز به لغو کار داشته باشیم. لذا از نظر تئوری باید در هر وضعیت، امکان لغو داشته باشیم. از طرف دیگر امکان اینکه از هر وضعیت یک وظیفه منتهی به لغو پیاده سازی کنیم وجود ندارد چون دیاگرام بسیار پیچیده و شلوغ شده و مشکلات عدیده ای را ایجاد خواهد کرد. نکته دیگر این که این نوع از لغوها موردی و استثنایی است و به صرفه نیست که برای این موارد، دیاگرام و فرایند را تا این حد شلوغ و پیچیده کنیم. بنابراین برای این مشکل، راه حل تغییر وضعیت از طریق پیام را انتخاب می کنیم.

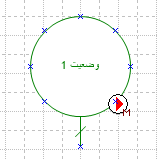
### مراحل پیاده سازی تغییر وضعیت از طریق پیام

1. یک وضعیت آغاز پیام در دیاگرام درج کنید:

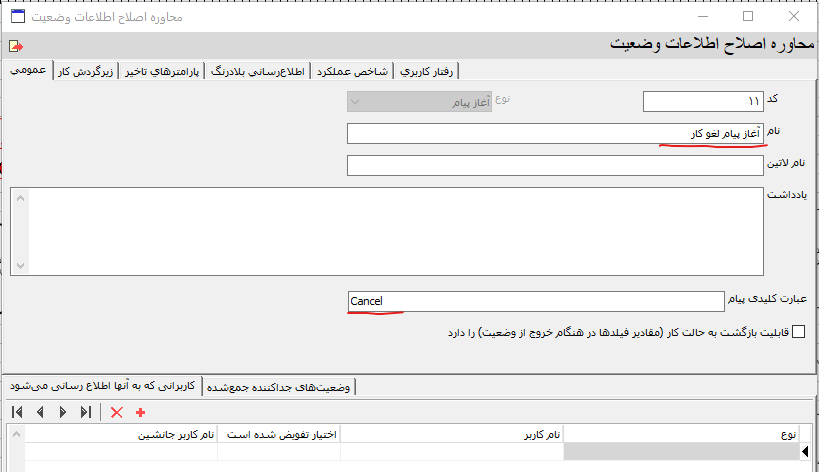
وضعیت آغاز پیام یک وضعیت تصمیم گیر است که با حرف M متمایز شده است. این وضعیت در نوار وضعیتهای دیاگرام، با فلش قرمزرنگ مشخص شده است.



به این ترتیب، یک وضعیت به شکل زیر در دیاگرام ایجاد می شود:



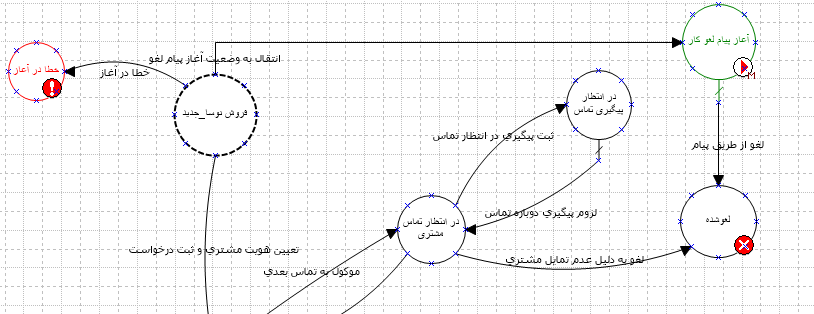
1. حال روی این وضعیت دبل کلیک کنید تا محاوره اصلاح اطلاعات وضعیت به شکل زیر باز شود:



همانند شکل بالا، در این محاوره و در قسمت نام، نام دلخواه خود را برای این وضعیت وارد کنید (در اینجا "آغاز پیام لغو کار") و ضمنا در قسمت عبارت کلیدی پیام، یک عبارت لاتین مناسب (در این مثال Cancel) را وارد کنید.

1. یک وظیفه از وضعیت شروع به وضعیت آغاز پیام ایجاد می کنیم.
2. یک وظیفه از وضعیت آغاز پیام به وضعیت مورد نظر (در اینجا وضعیت لغو) رسم می­کنیم.

به شکل نهایی این در مثال دیاگرام فروش (شکل زیر) توجه کنید:



**توضیح:** وظایف ورودی و خروجی از وضعیت آغاز پیام مثل هر وظیفه دیگر می توانند شامل توابع اجرایی باشند. ضمنا وضعیت آغاز پیام همانطور که قبلا گفته شد، از نوع تصمیم گیر است و لذا می تواند بیش از یک خروجی داشته باشد و براساس شرایط، به وضعیتهای مختلف منتقل شود و حتی ممکن است پس از گذر از ترکیبی از وضعیتهای متعدد به یک وضعیت دلخواه منتقل شود.

## تکمیل مراحل پیام با درنظر گرفتن وضعیتهای آغاز پیام

مراحل پیام قبلا و بدون توجه به پیامهای تغییر وضعیت توضیح داده شد اما حال که با این نوع از پیامها آشنا شدیم بهتر است با مراحل پیام بطور کامل آشنا شوید:

**مرحله 1)** ارسال پیام به شکل دستی از طریق رابط کاربری یا از طریق برنامه و توسط تابع WorkSendMessage

**مرحله 2)** قرار گرفتن پیام در لیست انتظار

**مرحله 3)** تحویل پیام به کار مورد نظر (گیرنده)

**مرحله 4)** اجرای تابع WorkProcessMessage

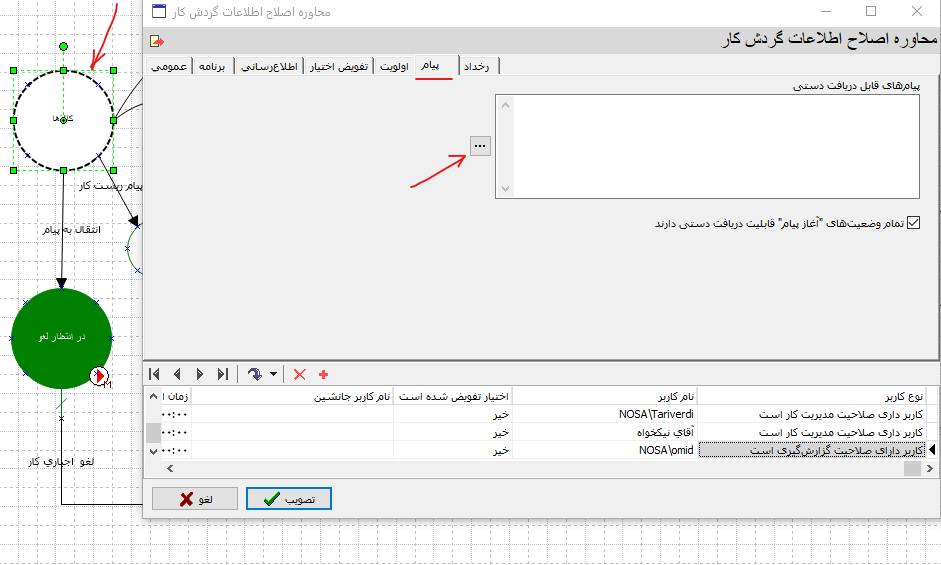
**مرحله 5)** اجرای وظیفه "انتقال به وضعیت آغاز پیام"

**مرحله 6)** اجرای وظیفه خروجی از وضعیت آغاز پیام

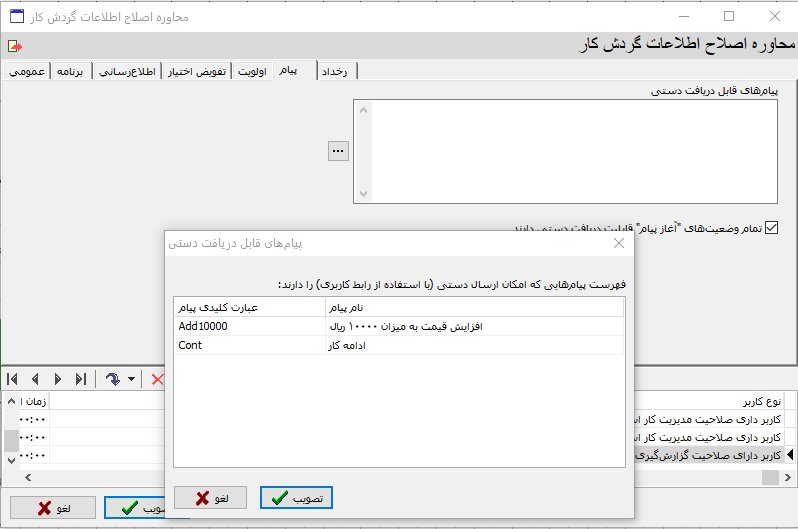
**توضیح:** بسته به پیام، ممکن است بعضی از مراحل شش گانه اجرا نشود برای مثال اگر پیامها به شکل دستی و با استفاده از رابط کاربری ارسال شود، آنگاه مرحله اول (ارسال با تابع WorkSendMessage) وجود ندارد. یا مثلا اگر تغییر وضعیتی قرار نیست رخ بدهد آنگاه مراحل 5 و 6 اجرا نمی­شود.

# ضمیمه 1: معرفی پیامهای قابل دریافت دستی به گردش کار

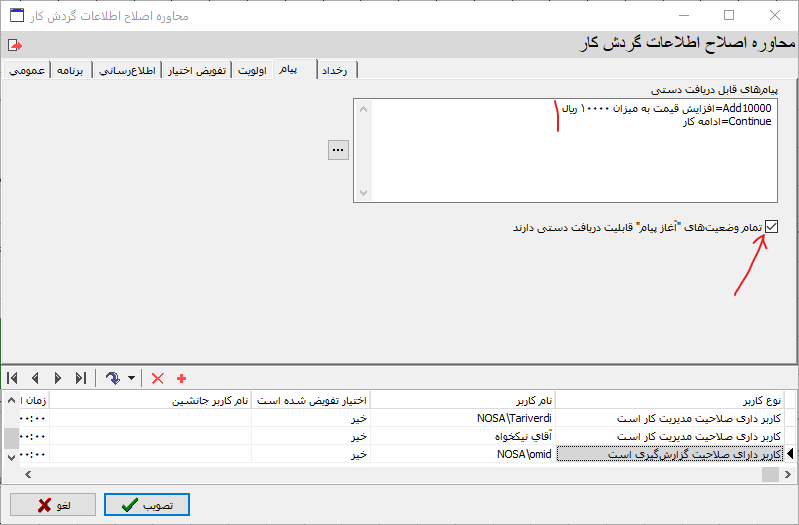
همانطور که قبلا توضیح داده شد، یکی از روشهای ارسال پیام، به شکل دستی و با استفاده از رابط کاربری است. برای این کار، ابتدا باید پیامهای قابل دریافت دستی را به گردش کار معرفی کنید. به این منظور روی وضعیت شروع گردش کار (در شکل زیر با فلش مشخص شده) دبل کلیک کنید تا محاوره اصلاح اطلاعات گردش کار باز شود و از آنجا روی زبانه "پیام" کلیک کنید تا محاوره به شکل زیر روی صفحه باز شود:



حال در قسمت پیامهای قابل دریافت دستی، تکمه  را کلیک کنید تا محاوره تعریف پیامها به شکل زیر باز شود:



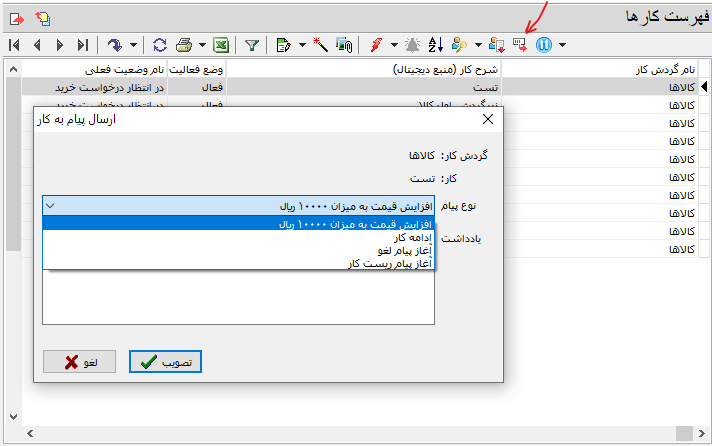
مطابق شکل بالا، فهرست پیامهای قابل دریافت دستی را وارد کنید و در انتها تکمه تصویب را کلیک کنید. به این ترتیب فهرست پیامها را به شکل زیر مشاهده خواهید کرد:



**نکته مهم:** تمام پیامهای معرفی شده در این سیستم باید قبلا در تابع WorkProcessMessage معرفی و عملکرد آن مشخص شده باشد. به عبارت دیگر گزینه های این فهرست در واقع از عبارتهای کلیدی پیامهای تعریف شده در تابع WorkProcessMessage استخراج می شود و باید عینا مشابه آنها باشد، در غیر این صورت،پس از ارسال پیام با اینکه پیغام "ارسال پیام به کار با موفقیت انجام شد" ظاهر می شود اما در واقعیت هیچ کاری انجام نمی شود و پیام بدون تاثیر خواهد بود.

**گزینه تمام وضعیت های "آغاز پیام" قابلیت دریافت دستی دارند**: این گزینه در حالت پیش فرض فعال است و به این معنی است که تمام پیامهای تغییر وضعیت، امکان دریافت دستی دارند و دیگر نیازی نیست که این پیامها را در بخش پیامهای قابل دریافت دستی اضافه کنید.

حال از این پس اگر در فهرست کارها روی گزینه پیام  کلیک کنید، این گزینه ها را مشاهده خواهید کرد:



ملاحظه می کنید که فهرست پیامهای دستی، چهار گزینه دارد که دو مورد اول در بخش پیامهای قابل دریافت دستی تعریف شد و دو مورد بعدی (آغاز پیام لغو و آغاز پیام ریست کار) از نوع پیامهای تغییر وضعیت هستند که شامل وضعیت آغاز پیام هستند و چون در محاوره قبلی، گزینه **تمام وضعیت آغاز پیام قابلیت دریافت دستی دارند** را تیک زده بودیم، این دو پیام نیز بطور اتوماتیک به فهرست پیامهای دستی اضافه شده است.