White Paper

Creating a modern CICS Web Application

Matter of Fact Software

Stephen Mitchell

2020

enquiries@matteroffactsoftware.com

www.matteroffactsoftware.com

Summary

Web interfacing with CICS is a topic that has fascinated some of us for many years. IBM made doing so possible with the introduction of the CICS WEB API extensions along with giving it native ability to communicate across TCPIP directly via TCPIPService and URI CICS Resource definitions.

The addition of these capabilities is a good thing, though IBM themselves do not appear to promote their usage. Perhaps this is because they are somewhat conflicted given they have an interest in many other technology sectors and don't wish to impact sales and marketing efforts elsewhere within their technology portfolio. Nevertheless, businesses that already have CICS at their disposal would be wise to seek ways to exploit these capabilities to their own advantage.

At this point it should be noted that gaining the business benefits of these features of CICS requires mounting a learning curve and investing time and effort in doing so. You will need to have motivated systems programmers and developers to climb that learning curve and all the while, there will be doubt and scepticism within your technology departments that anything worthwhile will come of such an investment of resources. However, in my opinion, this is well worthwhile.

Learning how to exploit these capabilities of CICS will germinate fresh ideas amongst your technical staff about how they can approach the resolution of technical challenges faced by your business. However, unless great care is taken at the outset, you will find yourself reinventing the wheel over and over again to bring solutions to fruition. This is because there are a number of different factors that have to be handled in order to build, manage and operate any and all CICS Web applications.

Matter of Fact Software, was founded to exploit these CICS Web interfacing capabilities by producing solutions within the DevOps arena. Our two products sit firmly in this arena:

PlexSpy, offering unique insights into named business application status allowing support staff to identify possible causes of operational issues faster than ever and usable with little or no mainframe expertise.

CICS Content Delivery Server (CICS CDS) was conceived whilst we were considering how to evolve PlexSpy. It came to life originally as CICS JS/Server but it evolved to its current form in which it provides capabilities to make it easy to create, manage and serve novel CICS Web applications.

This White Paper examines how it is now possible to approach the creation of a new CICS Web Application using **CICS Content Delivery Server** which, essentially, does much of the heavy lifting for you.

The Challenge

Make the IBM CICS Sample application, FILEA, available via a new Web Application.

Background

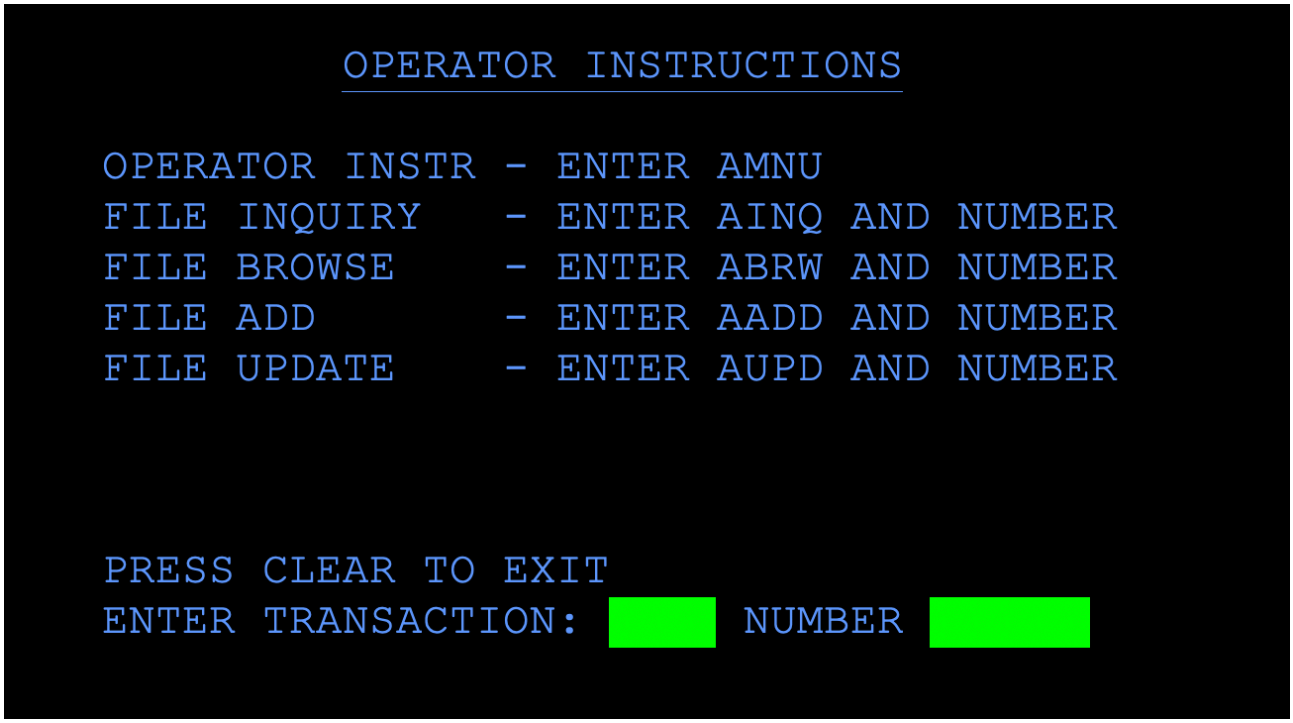
The FILEA sample application has been present in CICS for many years. It is a 3270 green screen driven application that accesses VSAM data housed with the FILEA file. This file can be defined and loaded with data by running a job provided by IBM in the SDFHINST dataset.

SDFHSAMP(DFH$FAIN) houses sequential records that are loaded into the file as part of the job held in SDFHINST(DFHDEFDS)

The CICS Transactions, programs and Mapsets required to run the original application are defined within CICS CSD
Group(DFH$AFLA). The FILEA file definition along with a couple of TDQueues are to be found in CSD Group(DFH$FILA). Adapt these definitions for your site and install them in a CICS region then use

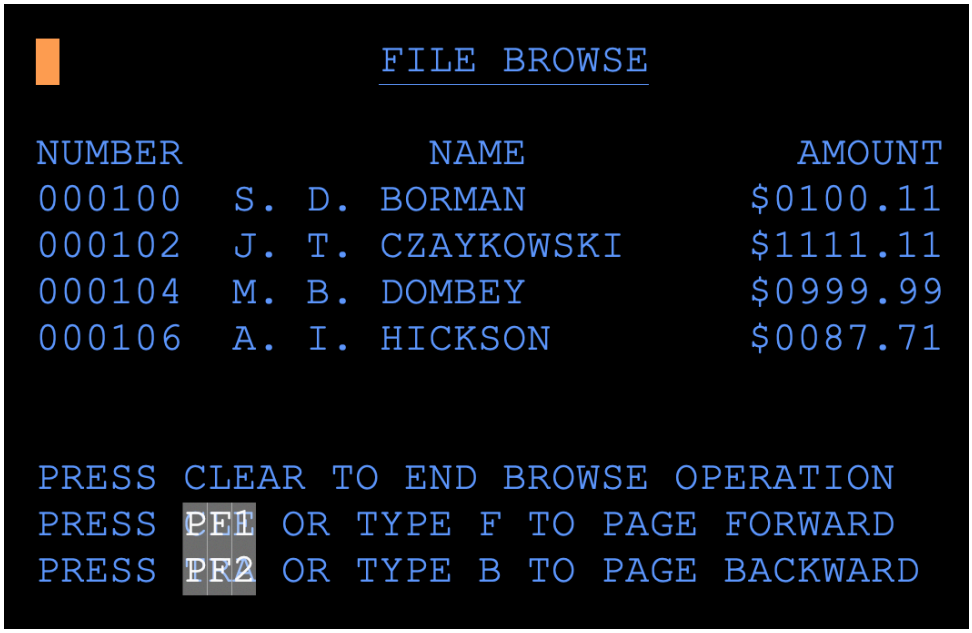
the **AMNU** transaction in a cleared CICS 3270 session to start the application Menu.

It should appear as follows:



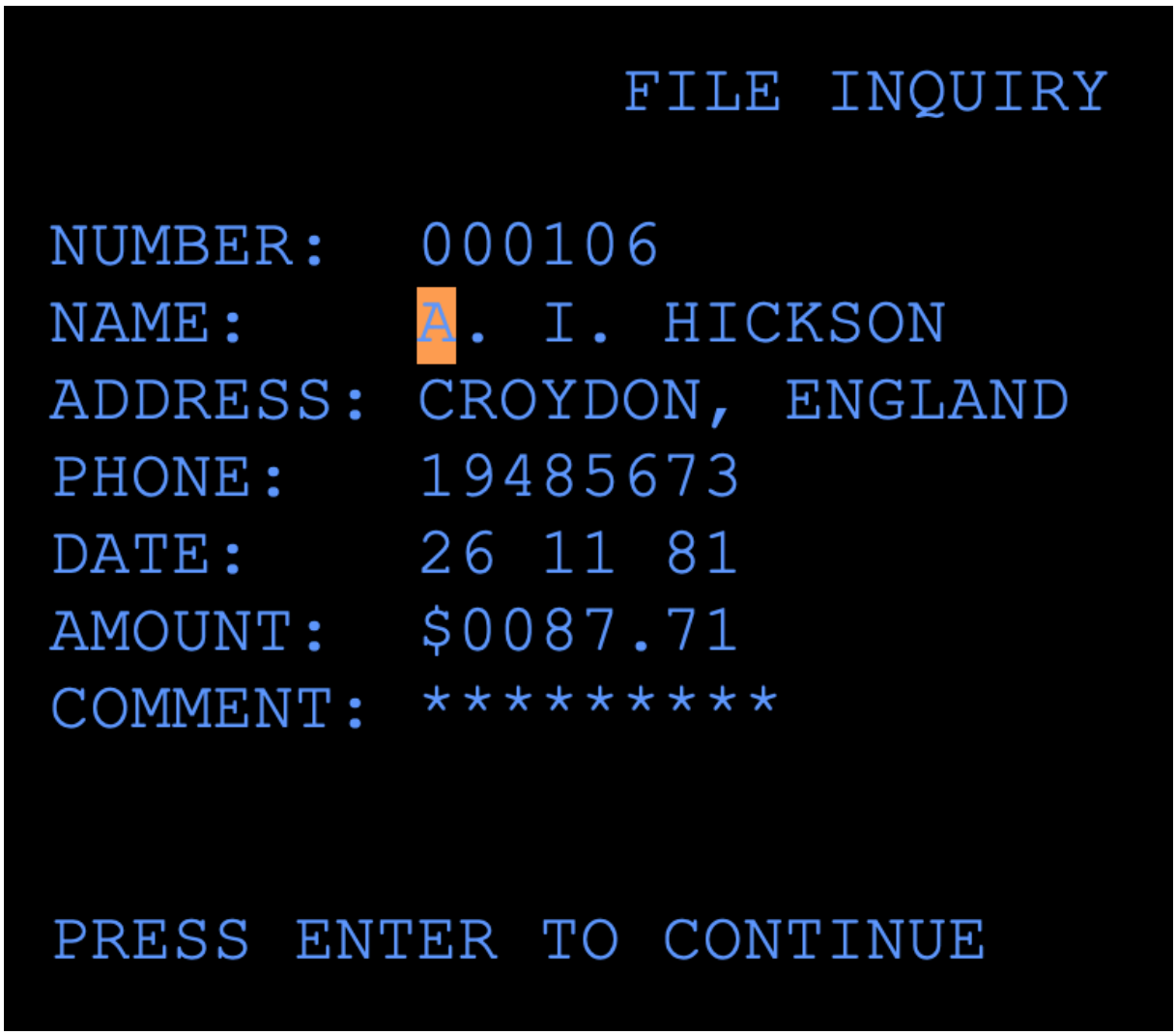
The input fields allow the file access transactions: AINQ; ABRW; AADD & AUPD can be used and, if required, a record Key number provided.

Entering **ABRW** in the transaction filed returns the following:



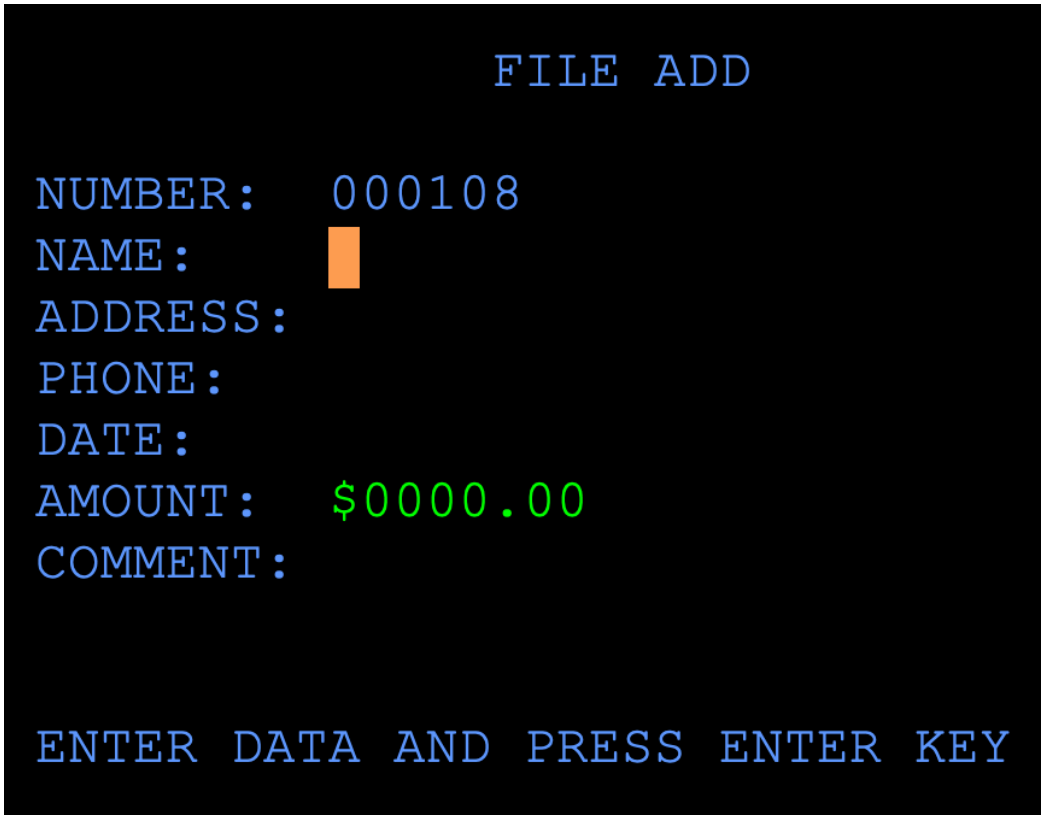
As can be seen, only four records at a time can be displayed and the user must use ‘F’ or ‘B’ for page forwards and backwards through the file records.

Using **AINQ** and a valid record Key value will return a display of the record as follows:



Using **AUPD** will bring up a simpler display but will allow the user to overtype the data in the fields and then save it to the FILEA file.

Using **AADD** and providing a new (unused) record key will result in a display as follows:



Values can be added to the fields and hitting ENTER key should result in the new record being saved to the file.

As an application, the original sample is quite simplistic and not terribly user friendly but it does act as a sample of what can be done with CICS as it was all those years ago.

We decided it needed an update.

Our Approach

CICS CDS does much of the work needed to serve CICS Web Applications, leaving the developer free to focus attention on the actual needs of the application. Of course, we wanted to take advantage of those capabilities to achieve our aims and it offers a number of ways of tackling the task.

Following a simple installation process, we implemented and configured CICS CDS to run in one of our development & test CICS regions. From that point we had a platform to develop our new application, taking advantage of the powerful features provided by the software.

We chose to build the front end using *css grid elements* to provide a consistent and flexible page layout and to use an AJAX call to a CICS program from within JavaScript to fetch the data from FILEA. Other JavaScript elements were written to construct the data displayed and to enable the various record management processes (Update / Add / Browse / Delete / Read).

Given that the FILEA normally only has a small number of records in it, we decided to pre-load *all* of the records on file into the browser page as the application starts. These are displayed a selectable list in the left hand side panel of the html page.

Obviously we needed a new program that can respond to Ajax calls from the browser and this we called SAM0FILA. We chose to write it in Cobol. The program was written to interact with CICS CDS, accepting a Channel/Container which references the web response (document) being constructed by the application. The program uses CICS WEB READ FORMFIELD commands to retrieve action pairs sent to it from the web Browser.

The Back-End - CICS Cobol Program

SAM0FILA has been supplied in both source and load module form. It can be called via an AJAX call from the front-end web application.

The program has been written to function as a CICS CDS program and storage areas are included in its STORAGE SECTION to fulfil this requirement. A COBOL COPYBOOK, MF3LNKAG, is included in the WORKING-STORAGE SECTION.

In its PROCEDURE DIVISION, the program looks for a CHANNEL.

An **EXEC CICS ASSIGN CHANNEL** command must return a value: **MF3-CHANNEL** to qualify. If this is found, it then then issues:

```
EXEC CICS GET CONTAINER
 CONTAINER ('MF3-DTOKEN      ')
 CHANNEL ('MF3-CHANNEL     ')
 INTO ('................................')
 FLENGTH (32)
```

The 32 byte **INTO** area contains two sixteen byte document tokens that have been created by the CICD CDS framework. The first of these is the document token for the web document being built and the second containing symbols if these have been used.

These can be used if the program needs to insert content into the web document that will be returned to the caller.

After initial processing, the program will interrogate the web form data that should have been sent to it. It passes through a section of code that retrieves the various items of form field data that it expects and IT then does what has been requested of it in those form fields.

The first of these form fields , "*call*" is the function requested which can only be one of the following:

“*browse*”, “*read*”, “*add*”, “*update*” or “*delete*”.

The program responds to a *browse* request by browsing all records in the FILEA file and building a JSON data response containing all of that data.

Similarly, it responds to a *read* request by reading a specific record using the record *key* supplied to it in the form field data. The record field values are returned to the caller using JSON data which is written into the web document.

add a record requires form field data for every field in the record to be present in the supplied form field data. The new record is constructed from this data and written to FILEA. A confirmation is returned to the caller using JSON data.

delete requires the record *key* to be deleted to be present in the form field data passed to the program. The *key* is used to read the record and it is then it is deleted. A confirmation is returned to the caller via JSON data.

An *update* request will overwrite an existing record with the values provided in the input form fields. This requires requires the record *key* to be provided along with all record field values. A confirmation is returned to the caller via JSON data following a successful rewrite command.

In this case, the SAM0FILA program does not issue a CICS WEB SEND command, instead it simply returns to its caller (a CICS CDS framework program) which will issue the CICS WEB SEND on behalf of the application.

The Front-End

We wanted to serve every part of our application from CICS on mainframe z/OS using CICS CDS.

Firstly, we had to chose a CICS CDS Repository to house the application. Each repository is a VSAM file that has been defined and initialised and then the instance of CICS CDS to be used has a configuration parameter added to enable its usage. We could have defined a new file at this point but decided to use the SAM file as it was already in place. We could chose to move or copy the application artefacts to another file later if required, in any case.

Next, we needed to decide a web path for our application. We decided to use the following knowing that we could make that unique:

/sam/application/filea/

So, assuming the application will be driven from a file called filea.html, our full application URL will be:

/sam/application/filea/filea.html

OK, we are ready to start using CICS CDS to define the application.

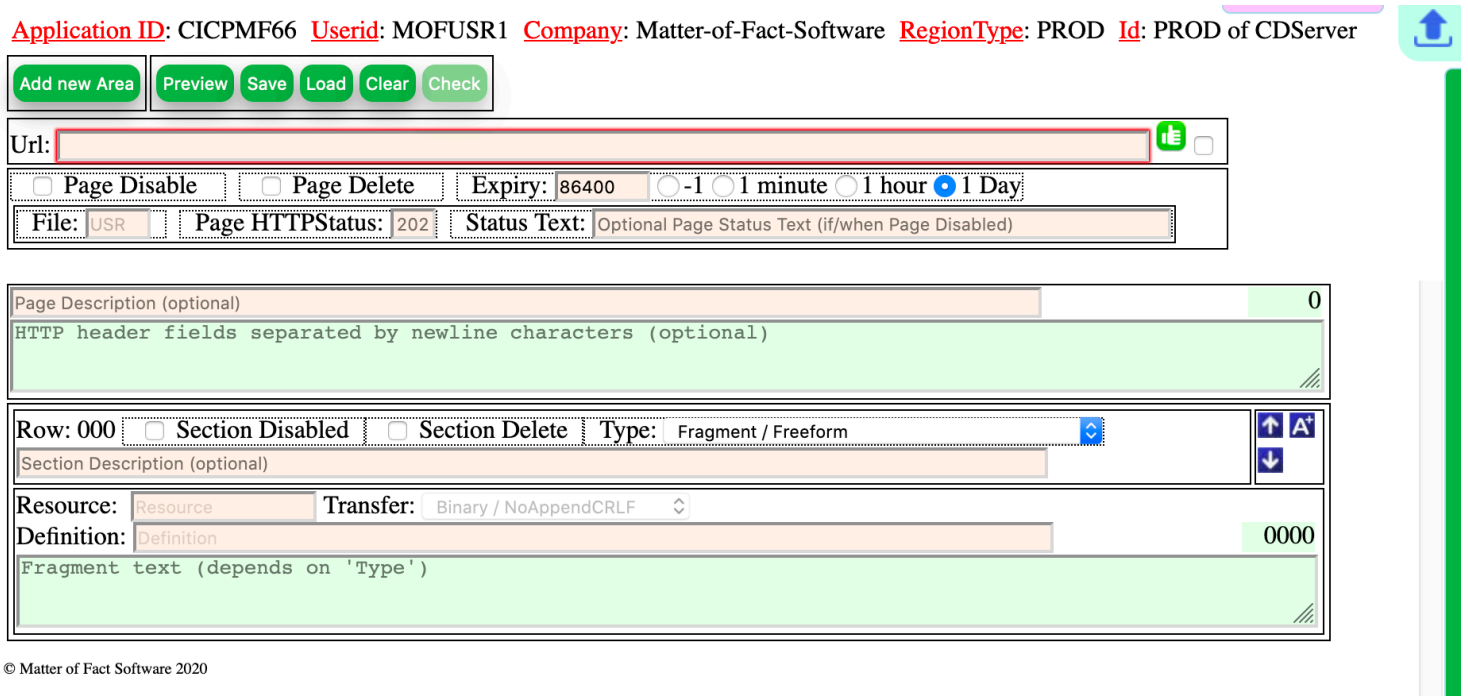
In the first instance, we need to start up the **CICS CDS Editor**.

This can be done by targeting the URL for the Editor in a Web Browser as follows:

https://777.888.999.000:9566/CICS/MF3X/mf3/mf3contentu/basicTextEntry.html

https is needed if your site has SSL enabled (recommended). The url address can be the actual numerical address or a dns-name. The port number must be the one defined to the CICS CDS instance as a TCPIPService (MF3ADMIN). In our case, this is shown as 9566.

Once the Editor is loaded into the Web Browser, it appears as shown below:



The chosen URL for the new application will need to be typed (or pasted) into the URL field on the web-form. Clicking away from that field will cause CICS CDS to verify the chosen URL at this point, so you will be warned if it already exists on file - the URL being used is part of the record key. As our url begins /sam/ , it is the MF3#SAM file that forms the repository for the application.

It is advisable to create your new applications html, css and JavaScript code using whatever off-host editor that you have at your disposal.

The front end code can be developed and verified using static data to reach a point at which you want to transfer it to the mainframe.

That was the approach we took and we ended up with a number of artefacts as follows:

filea.html
/css/fileas.css
/js/filea.js
/images/favicon.png
/images/newadd1.png
/images/reload1.png
/images/clear1.png
/images/help1.png
/images/idelete1.png
/images/iupdate1.png
/images/iadduse1.png
/images/newLoad1.png
/json/filea-help.json

Each of these elements needed to be defined to CICS CDS in its own right. The .html, .css and .js files are initially created using the CICS CDS Editor.

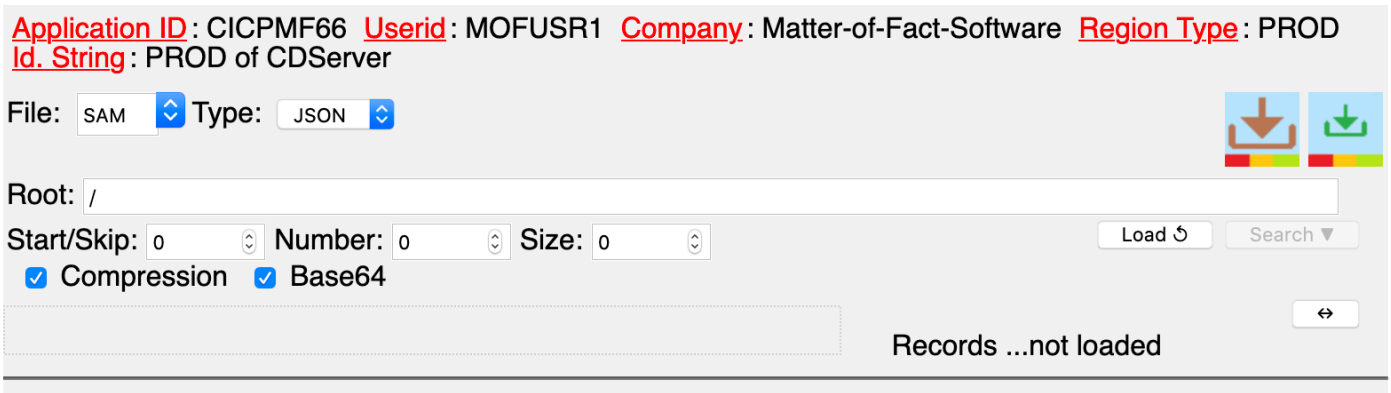
The code created being entered into the Freeform Text area on the Editor form along with its Url. Once saved, it can be executed either by typing its URL into a web browser address bar or from the CICS CDS Directory facility.

The Directory facility will display all or a subset of the records present in a given CICS CDS Repository file.

To start the Directory Facility, enter the following into a Web Browser Address bar:

https://777.888.999.000:9566/CICS/MF3L/mf3/directory/dir4.html

It appears as shown below:

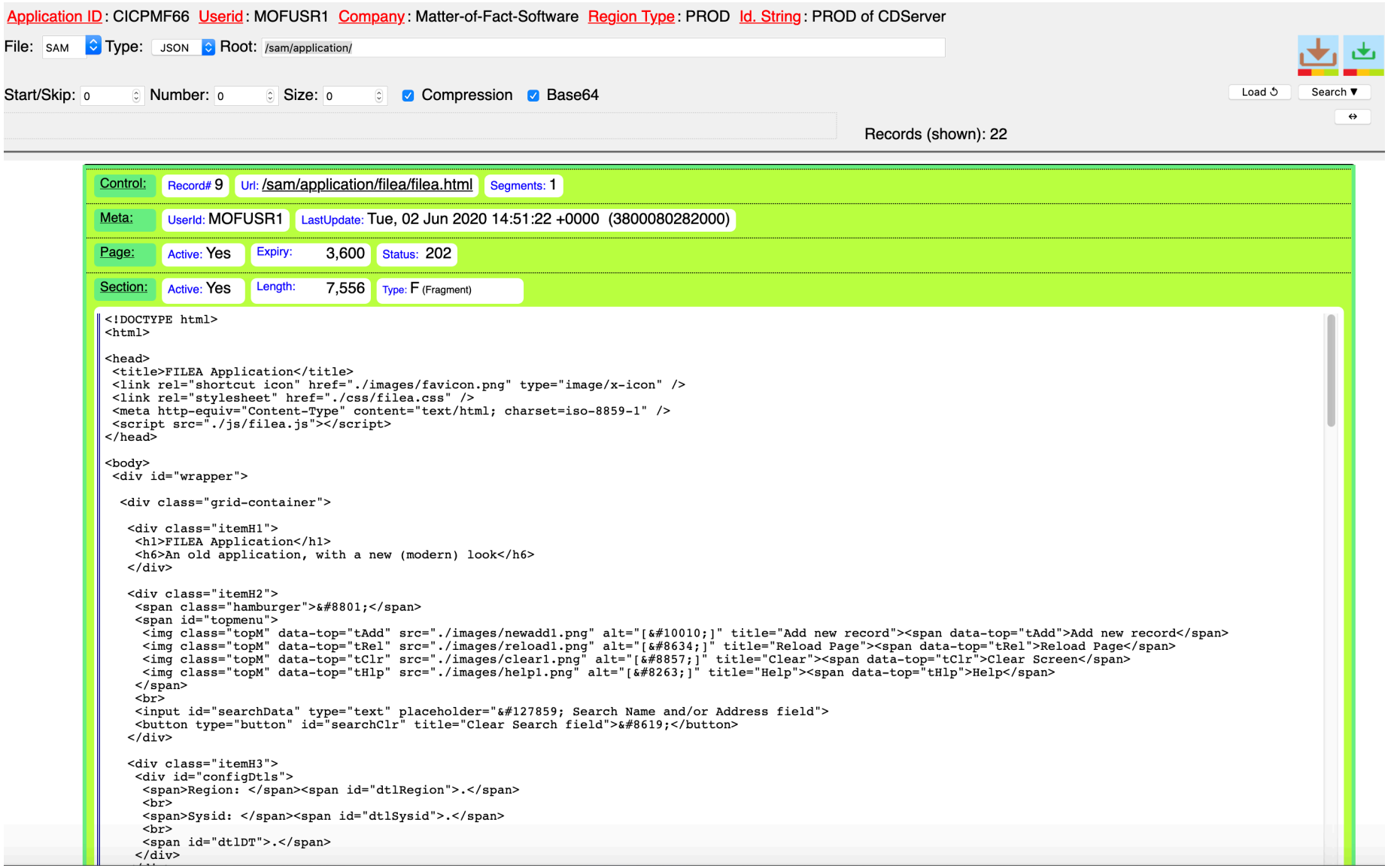


This empty Directory function is primed to load from the SAM file as can be seen in the ‘File’ input field setting. That dropdown selection box can be adjusted by the user if necessary. Leaving the File selection as ‘SAM’ and clicking the (Load) button will cause the CICS server to be called and ALL records in the SAM file will be displayed in the page below the form.

If the full record url is known, it can be typed or pasted into the Root input field. Clicking (Load) will then return that record detail alone. A partial url can also be entered in the Root field and that would return all records matching that url prefix.

The green bar on the right hand side of the display can also be useful here. Hovering the mouse pointer over it will cause it to pop-out, displaying another input field along with all of the records currently loaded in summary form (the url only).

Let’s assume we have only asked CICS CDS Directory function to display the filea.html record. It would appear in the display as shown overleaf:


Application ID : CICPMF66 Userid : MOFUSR1 Company : Matter-of-Fact-Software Region Type : PROD Id. String : PROD of CDServer
File: SAM Type: JSON Root: /sam/application/
Start/Skip: 0 Number: 0 Size: 0 Compression Base64
Load Search
Records (shown): 22
Control: Record# 9 Url: /sam/application/filea/filea.html Segments: 1
Meta: UserId: MOFUSR1 LastUpdate: Tue, 02 Jun 2020 14:51:22 +0000 (3800080282000)
Page: Active: Yes Expiry: 3,600 Status: 202
Section: Active: Yes Length: 7,556 Type: F (Fragment)
<!DOCTYPE html>
<html>
<head>
<title>FILEA Application</title>
<link rel="shortcut icon" href="./images/favicon.png" type="image/x-icon" />
<link rel="stylesheet" href="./css/filea.css" />
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<script src="./js/filea.js"></script>
</head>
<body>

<h1>FILEA Application</h1>
<h6>An old application, with a new (modern) look</h6>

≡

<img class="topM" data-top="tAdd" src="./images/newadd1.png" alt="[✚]" title="Add new record">Add new record
<img class="topM" data-top="tRel" src="./images/reload1.png" alt="[↺]" title="Reload Page">Reload Page
<img class="topM" data-top="tClr" src="./images/clear1.png" alt="[⊙]" title="Clear">Clear Screen
<img class="topM" data-top="tHlp" src="./images/help1.png" alt="[⁇]" title="Help">Help

<input id="searchData" type="text" placeholder="🍳 Search Name and/or Address field">
<button type="button" id="searchClr" title="Clear Search field">↫</button>

Region: .

Sysid: .

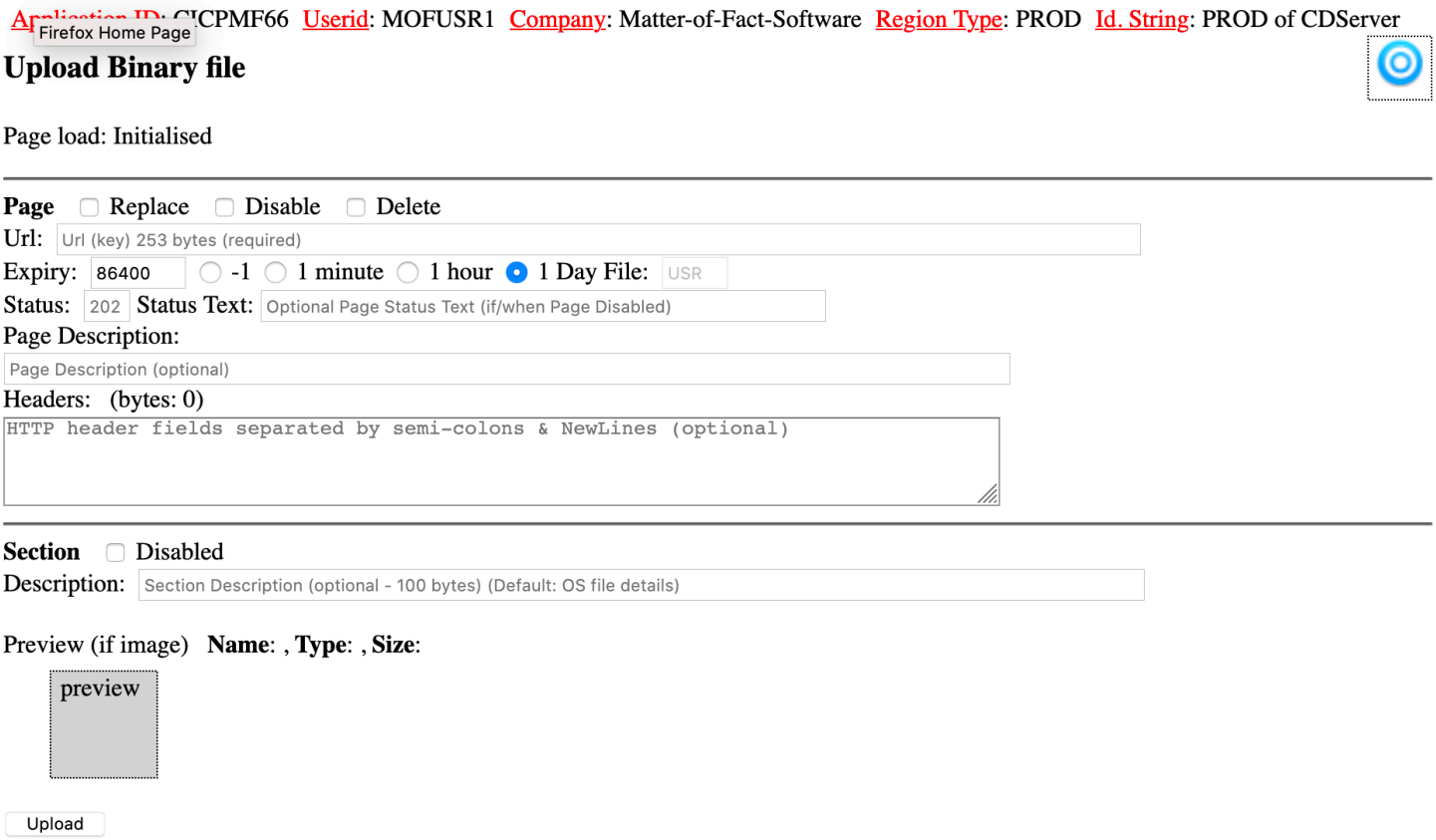
.

The image files (.png) referenced within the html are created using the CICS CDS Binary Upload feature.

To start the CICS CDS Binary Upload facility, use the following URL in a Web Browser:

https://777.888.999.000:9566/CICS/mf3e/mf3/editor/advanced/uploadbinary.html

Once loaded, the web page should appear as follows:



A CICS CDS record has a usable payload of approximately 31,900 bytes. Many of the files required to build a web application will be smaller than this but, should the need arise, records can be linked together to form a whole. In this way, CICS CDS gets around the 32K limit that would otherwise apply. Additionally, it is possible to take a larger off-host text file (html, css, js etc) and upload it using the Binary Upload feature when it will be automatically split up into what are termed Blocked records - these are records that will be treated as a single entity when served but are in fact a set of linked records on file.

Our css file is only 3.6K and fits easily in a single record.

The html file is only 7.5K.

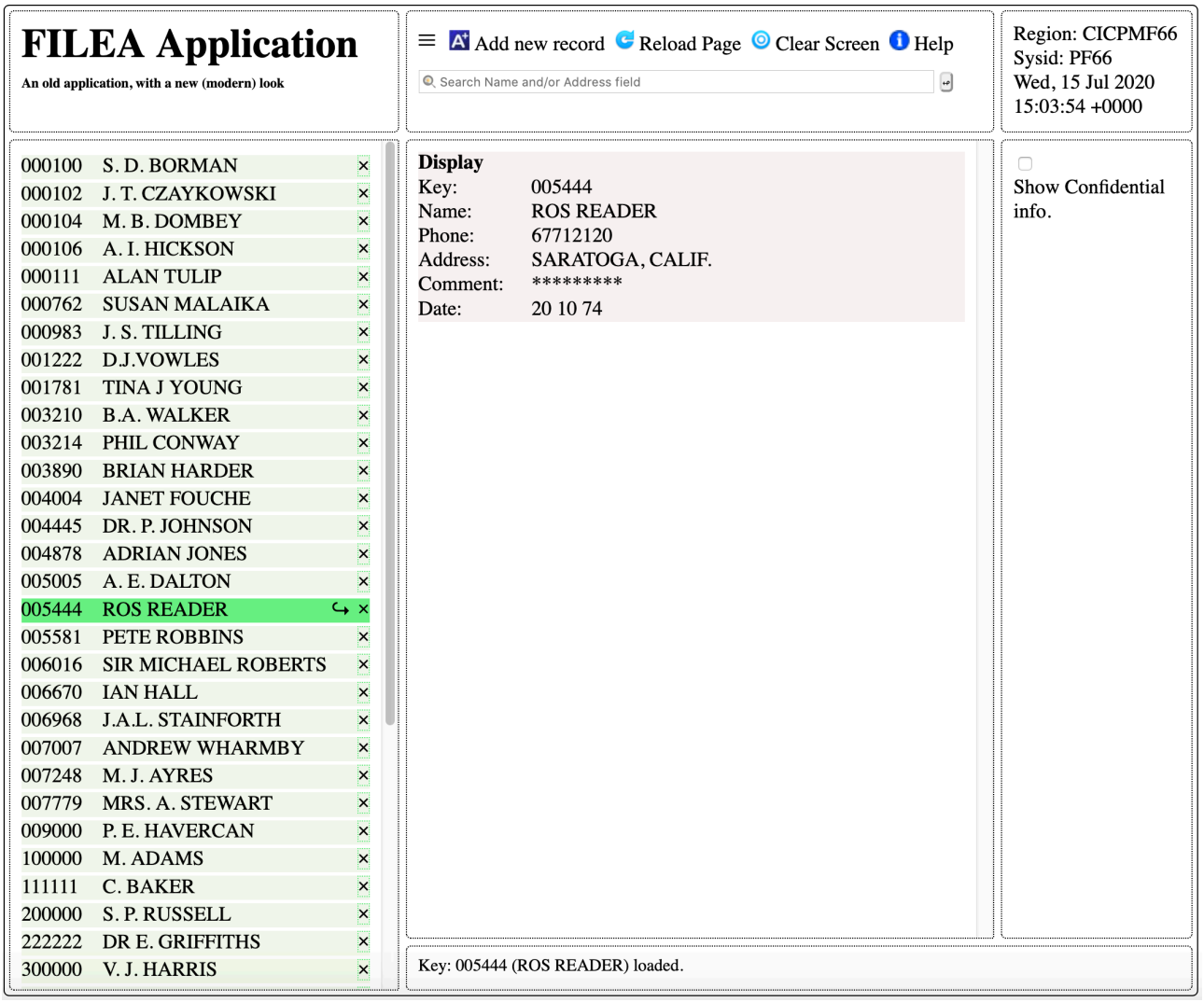
The js file is 30.5K and still (just) fits into a single record.

The image png files are all quite small so they do not need to be split up by CICS CDS.

The image file can be dragged and dropped onto the hot-spot on the right at the top of the page or by clicking on that hot-spot, a file open dialogue can be used to chose the image file required from your file system. The Url field needs a valid value. We chose */sam/application/filea/images/filename.png*

Being image files, you probably want to allow the default http cacheing header (Expiry) setting of 1 day but you can adjust that should you wish to do so.

Once all of the files have been successfully created, it is possible to point a web browser at the main html page and it will run. If the Cobol program is successfully called, then the data from the FILEA file will be loaded into the page as shown overleaf:


FILEA Application
An old application, with a new (modern) look
Add new record
Reload Page
Clear Screen
Help
Search Name and/or Address field
Region: CICPMF66
Sysid: PF66
Wed, 15 Jul 2020
15:03:54 +0000
000100 S. D. BORMAN
000102 J. T. CZAYKOWSKI
000104 M. B. DOMBEY
000106 A. I. HICKSON
000111 ALAN TULIP
000762 SUSAN MALAIKA
000983 J. S. TILLING
001222 D.J.VOWLES
001781 TINA J YOUNG
003210 B.A. WALKER
003214 PHIL CONWAY
003890 BRIAN HARDER
004004 JANET FOUCHE
004445 DR. P. JOHNSON
004878 ADRIAN JONES
005005 A. E. DALTON
005444 ROS READER
005581 PETE ROBBINS
006016 SIR MICHAEL ROBERTS
006670 IAN HALL
006968 J.A.L. STAINFORTH
007007 ANDREW WHARMBY
007248 M. J. AYRES
007779 MRS. A. STEWART
009000 P. E. HAVERCAN
100000 M. ADAMS
111111 C. BAKER
200000 S. P. RUSSELL
222222 DR E. GRIFFITHS
300000 V. J. HARRIS
Display
Key: 005444
Name: ROS READER
Phone: 67712120
Address: SARATOGA, CALIF.
Comment: *********
Date: 20 10 74
Show Confidential info.
Key: 005444 (ROS READER) loaded.

In the above, record number 005444 has been selected and the data from that record populated into the central display area.

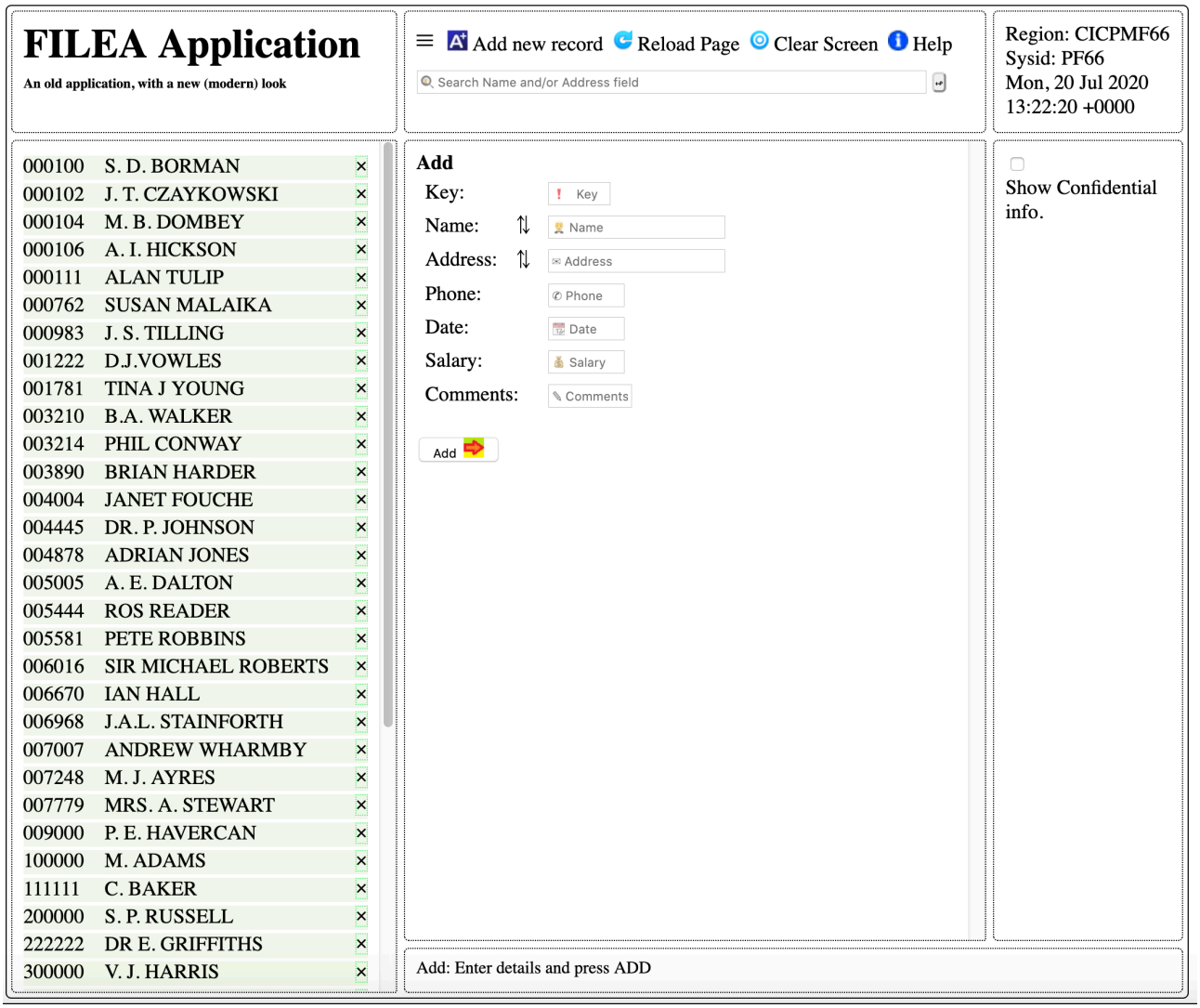
The slide bar allows the user to move up and down the list of available records to find the one required.

In the top right hand side of the page is an area showing the details of the CICS region in which the application is running along with date and time information.

The top middle grid element shows a search field. Entering a partial or full name here will result in the records in the left hand side section being reduced to only show those that match the search text entered. Search is done using JavaScript REGEX rules, so can be complex.

Also in the top middle section are a number of clickable icons:

 **Add a new record** will produce the following form within the middle display panel:



The user can enter values into the fields as required. Pop-up hints will appear for each field when hovering the pointer over any given field and form field validation is done before the form can be submitted to add the record to the FILEA file.

Reload Page will cause the file records to be refreshed from the FILEA on the host and re-populate the left hand panel.

Clear Screen will empty the central panel of all text and data.

Help will display information into the central panel providing a guide to using this application. It should be noted that the

help Information returned is itself a call for JSON data that is loaded from the following URL:

/sam/application/filea/json/filea-help.json

Suitable help text is provided with the sample application within the installed CICS CDS product. However, this can be changed using the editor if required.

On the right hand side of the display is a checkbox 'show confidential information'. If this is checked then the Salary data field from the records selected is shown. If it is not checked, then the Salary data remains hidden.

The grid element at the bottom of the page gives an indication of what the last action / usage was for the application.

Within the list of records in the left hand side list is an '**X**' at the right of the record number and Name. If the user clicks on that '**X**' or if the line itself is left-clicked, then a pop-up menu appears offering the following options:

Delete: Clicking this will result in a confirmation pop-up appearing allowing the user to confirm a request to delete the record.

Update: Clicking this will open the record in the central panel with the updatable fields available to change the data. A Save button can be used there to re-write the record with any updates.

Add Using: Clicking this will open a form with all of the data for the record highlighted. The user can adjust the fields as required but must enter a new (unique) record key before clicking the Add New button to save a new record.

Reload this Record from Host: Does exactly what you would expect - it causes the application to re-read the chosen record from the FILEA file and display it in the central panel.

At the end of the search field is a small icon that when clicked will clear the search field data causing all of the records on file to be displayed once more.

Using the search facility will result in information showing how many records have been excluded by the using the search filter.

Searches are for all fields except salary.

The salary data is converted to ASCII on host and send encoded as base64 to aid confidentially. CICS is capable to encode data in a different CHARSET and send the data in-stream, assuming the application/client is able to handle any specialised coding

The sample application does minimal checking.

The Cobol source is provided along with CICS CDS product. Users of this code may wish to further enhance some capabilities of the code.

Conclusion

Using CICS Content Delivery Server made the task of creating a modern take on the FILEA sample application very simple.

Without CICS CDS, we would have needed to make all of the application facets available via some other means (DOCUMENT TEMPLATES, probably), have them uploaded to the mainframe, defined, installed in CICS and then write CICS programs to access these and to manage the creation, population and sending of the web document. To do all of that is possible but to do so in a bespoke manner would be a lot of work and require reinventing the wheel every time you wanted to innovate a new web application.

We were able to design the front end elsewhere and then easily move its elements into CICS CDS using its Editor and Binary Upload functions.

It is necessary to either find or develop your own skills in writing html, css and JavaScript itself to reproduce what we did with this sample application. Alternatively, one could make use of one or more of the in-built Open Source toolkits that come with CICS CDS. Perhaps they might simplify the JavaScript required in the front-end.

It would be simple to add other such Open Source offering to CICS CDS if required.

In any case, it is clear that CICS CDS does what we set out to do with it. Namely, to make creating novel CICS web applications very easy. It should also be clear that other data mining processes could be deployed within the CICS program called (DB2 etc).

What business challenges could you solve using it?

For more information or if you would like to explore the capabilities of CICS CDS further, please contact Matter of Fact Software using the following E-Mail address: enquiries@matteroffactsoftware.com