

Speedy Incentives from Blaise 5 Instruments

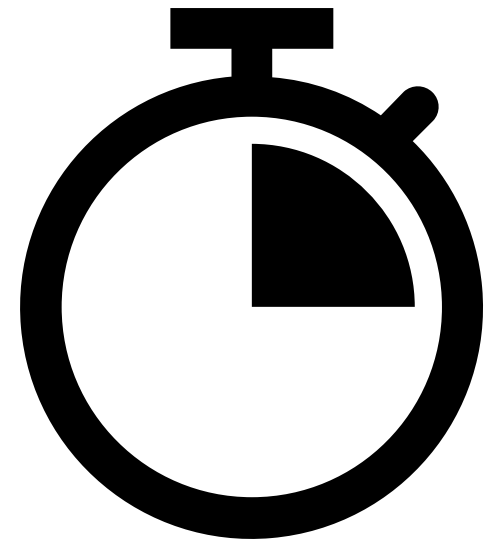
International Blaise Users Conference, October 2023

Emily Caron, Jerry Copperthwaite, and Rhymney Weidner - RTI International



Speedy Incentives

- Acknowledgements
- Introduction and Blaise Specifics
- Process overview
- Blaise Implementation
- Challenges and Lessons Learned
- Future Uses & Potential Enhancements
- Conclusions



Incentives

- Past RTI use and timing
- RTI initiative to implement speedy incentives for Blaise
 - Expansion of backend processes
 - Allow this to work for multiple data collection systems
 - Add texting capability

Blaise Instruments

- Household Screener and Selected R Survey
 - Both offered incentives
 - English and Spanish
- CAWI and CATI modes
 - Additionally, PAPI mode for Screener
- Blaise version 5.12.8



Process Overview

- Collect survey data using Blaise
 - Data stored in secure FIPS Moderate database (“FIPS Mod”, multi-factor authentication)



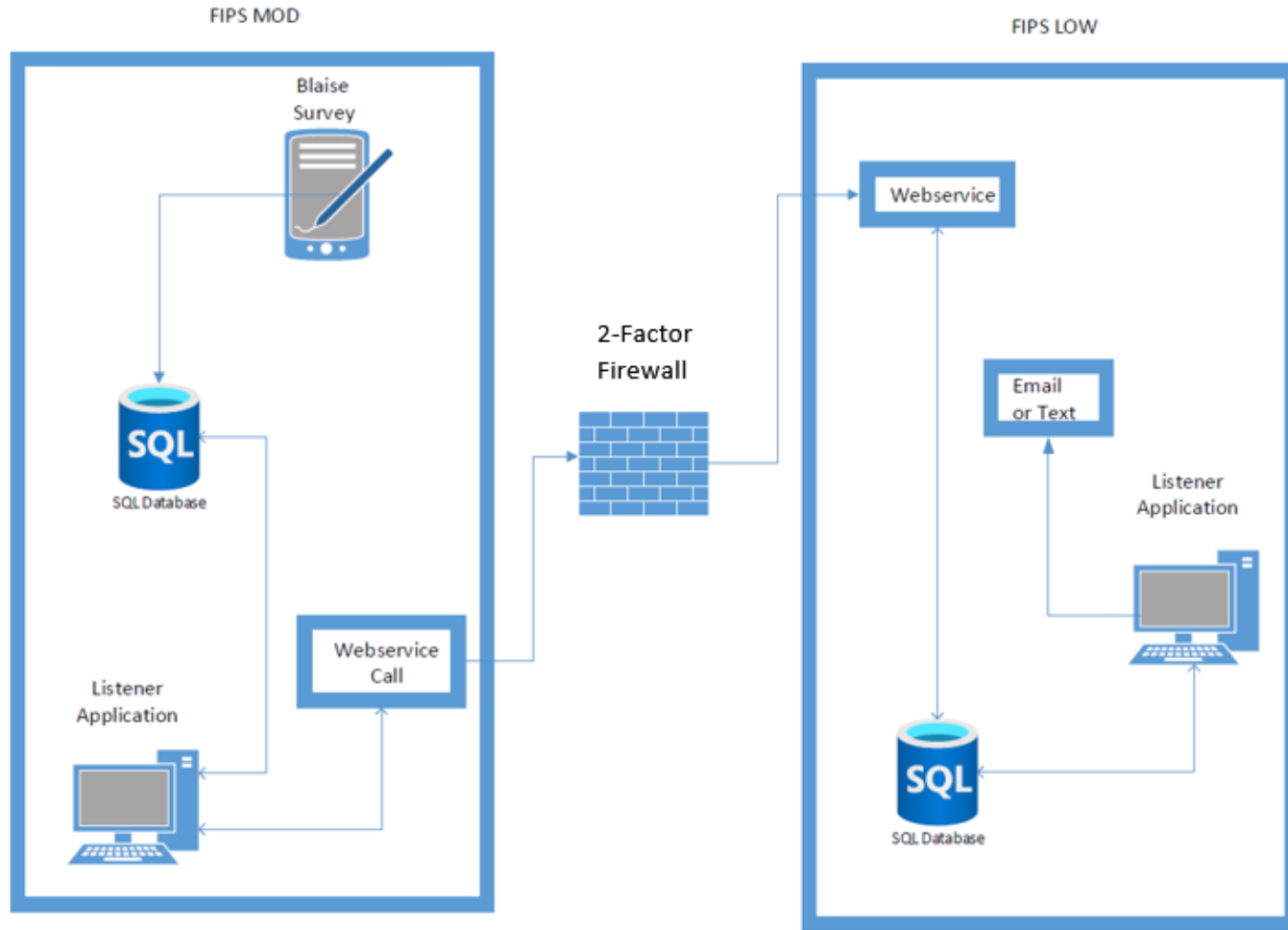
FIPS Mod

- Trigger writing incentive metadata to SQL table
- Listener application with timer
 - Queries SQL table for new records
- New record is found: call webservice

FIPS Low

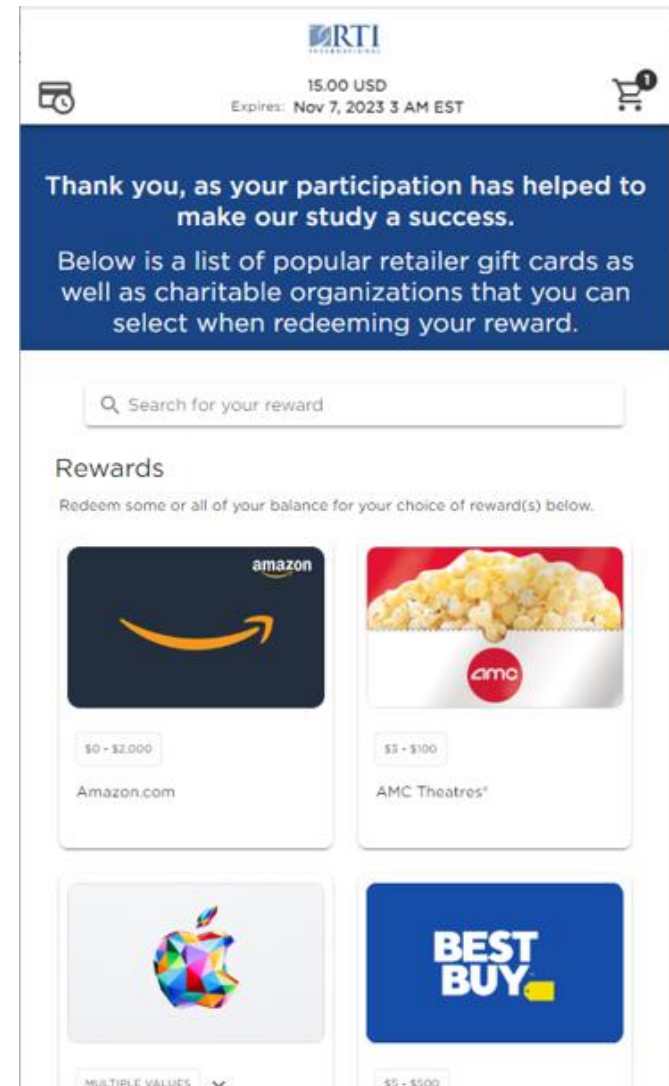
- Webservice does insert in SQL table
- Listener application with timer
 - Queries SQL table for new records
- New record is found: call incentive API to purchase new incentive
 - Resulting link is saved

Process Overview, cont.



Process Overview, cont.

- Build and send incentives
 - **Emails:** Simple Mail Transfer Protocol (SMTP)
 - **Texts:** ARTEMIS (in-house application)
- Incentives are handled through Tango®

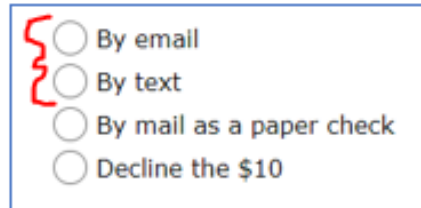


Implementation Questions

First, some important questions.

1. When do we want the incentives triggered?

- Only for digital incentives



A screenshot of a survey question with four radio button options. The options are: "By email", "By text", "By mail as a paper check", and "Decline the \$10". A red bracket highlights the first two options, "By email" and "By text".

- Send as soon as incentive-related questions are answered
 1. Collect email address or phone number
 2. Confirm delivery mode along with address or number entered
 - Consent provided
- Placement was near the end of the surveys, not necessarily at the end

2. How do we prevent duplicates?

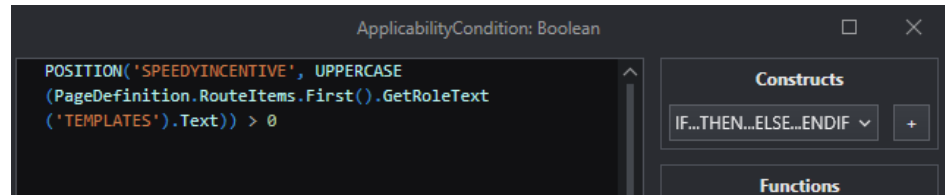
- Remove “back” button after process is triggered
- Fail-safe: backend processes also prevent duplicates

Blaise Implementation

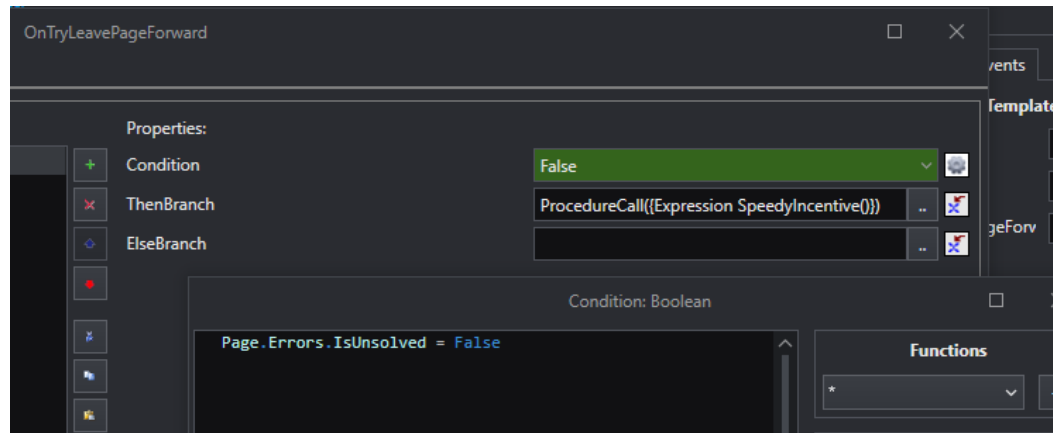
Special Template → Actions Setup function → PowerShell Script

- Active for email address/phone number confirmation screens
 - Applied to fields with “SPEEDYINCENTIVE” in Templates role

```
ENG "Just to confirm, would you like us to send  
<newline count=2>^{CellNumberWithFormatting  
SPA "Solo por confirmar, ¿quiere que le enviemos:  
<newline count=2>^{CellNumberWithFormatting  
TEMPLATES "SPEEDYINCENTIVE"
```



- Includes OnTryLeavePageForward event
 - Triggers after positive confirmation of email address or phone number

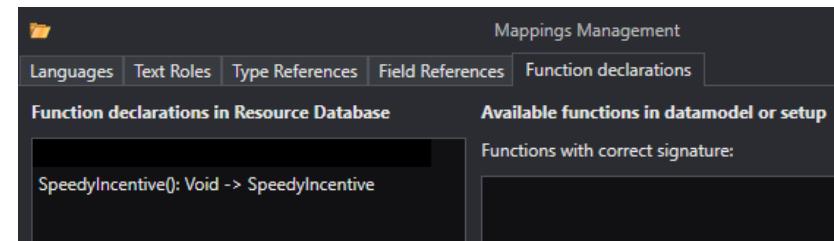


Blaise Implementation, cont.

Special Template → **Actions Setup function** → PowerShell Script

- “SpeedyIncentive” procedure is coded into an Actions Setup manipula
 - Specified in Project Properties

- Procedure is mapped in Mappings Management → Function declarations



- Relevant information is pulled from the active survey using SURVEYRECORD file definition

```
RULES
GUID := DepSession.GETINTERVIEWSTATE('InstrumentId')
CaseId := DepSession.GETVALUE('Main_Case.zrid')
//Add a letter to indicate language A=ENG, B=SPA
IF (DepSession.GETVALUE('main_case.SurveyLanguage') = 'SPA') THEN
    CaseId := CaseId + 'B'
ELSE
    CaseId := CaseId + 'A'
ENDIF
```

- Information is used to generate PowerShell scripts

```
IF UPPERCASE(SelectedMethod) = 'EMAIL' THEN
    //IF Email incentive
    strRun := 'PowerShell.exe -ExecutionPolicy Bypass -File "SpeedyIncentiveEmail.ps1" -caseid '"+CaseID+"' -email '"+Email+"' -fullname '"+ FullName+"' -blaiseid '"+ GUID + "' -amount '+ Amount
    aResult := RUN(strRun, WAIT, HIDE)

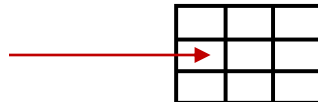
ELSEIF UPPERCASE(SelectedMethod) = 'TEXT' THEN
    //IF Text incentive
    strRun := 'PowerShell.exe -ExecutionPolicy Bypass -File "SpeedyIncentiveText.ps1" -caseid '"+CaseID+"' -phone '1'+ Cell +' -fullname '"+ FullName+"' -blaiseid '"+ GUID + "' -amount '+ Amount
    aResult := RUN(strRun, WAIT, HIDE)

ENDIF
```

Blaise Implementation, cont.

Special Template → Actions Setup function → **PowerShell Script**

- The PowerShell script calls a stored procedure
 - Numerous parameters passed through from the Actions Setup:
 - CaseID with language indicator
 - Confirmed email or cell phone number
 - Full name place holder
 - Blaise GUID
 - Incentive amount
- Stored procedure inserts row into SQL table
 - Listener application picks things up from there (see process overview)



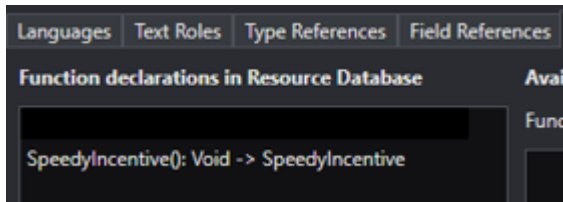
Challenges and Lessons Learned

- Error received when transitioning to FIPS Mod

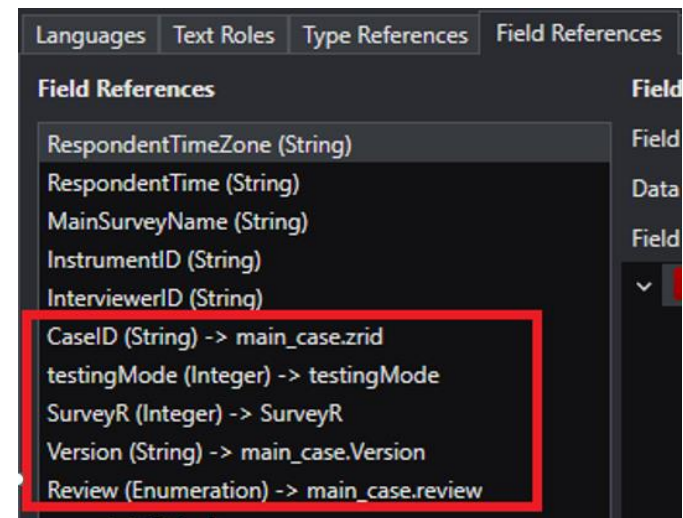
```
PS C:\InstallBlaise\ > .\test1.ps1
WARNING: A network-related or instance-specific error occurred while establishing a connection to SQL Server. The
server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured
to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)
PS C:\InstallBlaise\ > _
```

* Solution: open proper ports

- Blaise Mappings wouldn't "stick"



* No permanent solution, just kept checking on these and re-mapping when needed



Challenges and Lessons Learned, cont.

- Full process testing in production environment
 - In general: Don't allow test cases to receive real \$
 - Allow some test cases to go through



* Solution:

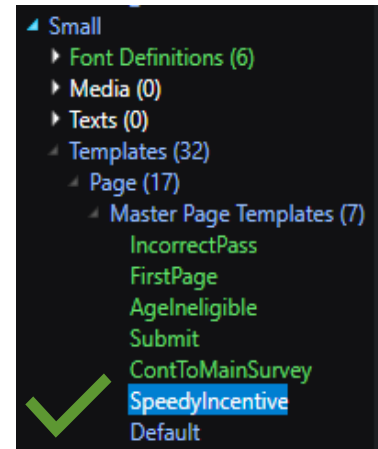
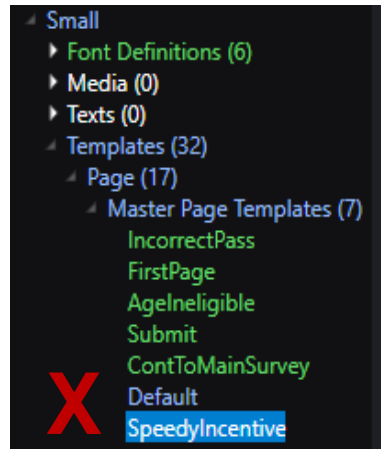
- PowerShell script update
- Carefully controlled testing

```
IF (($caseid -like "999*") -And ($caseid -notlike "9990462*") -And ($caseid -notlike "9990469*") -And ($
{
  $SqlCommand.CommandText = "EXEC CaseInsertTextTestBlaise '$caseid','$phone','$fullname', '$blaiseid', $amc
}
ELSE
{
  $SqlCommand.CommandText = "EXEC CaseInsertTextBlaise '$caseid','$phone','$fullname', '$blaiseid', $amount"
}
```

Challenges and Lessons Learned, cont.

Thoroughly test all layout sets

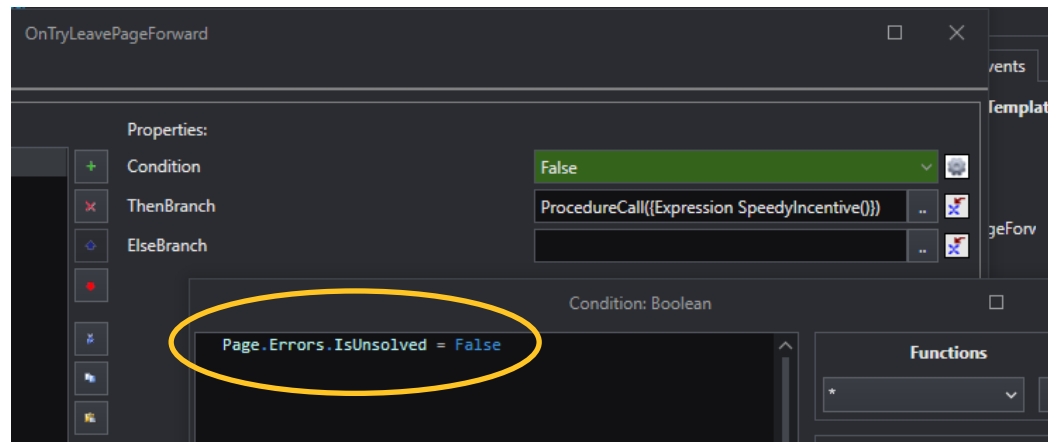
- Template hierarchy initially incorrect for small layout
- Early test efforts focused heavily on large screen layout (508)
- Revealed the large percentage using small devices to complete the survey (over 50% at the time of review)



Challenges and Lessons Learned, cont.

Carefully evaluate when events are executed

- OnTryLeavePageForward triggers as soon as the respondent *attempts* to leave the page
- When errors occurred on the page, the Speedy Incentives process was called twice
 - Duplicate entries in the db
 - Note: back-end processes prevented duplicate rewards sent
- Conditional statement was added to factor in unresolved errors



Future Uses and Potential Enhancements



Flexibility: The Speedy Incentives process can now handle multiple survey data collection systems

- Can serve a wide array of projects



Future projects can consider this option to reward respondents with a form of immediate gratification, whether via text or email, to encourage participation



Processes involved may one day be utilized to send other digital communications in a timely fashion

Possible examples:

Timely reminders to the respondent or others in the household

Handing a portion of a survey off to a co-worker as part of a business survey



Speedy incentives expansion provides us more options to encourage respondent participation



The results of this first implementation of Blaise and texting capabilities will be analyzed

Interesting comparison:
Survey component participation rates where screener R chose digital incentives vs. mailed or no incentive



Continuing to monitor... lessons learned will be shared and applied

More Information

delivering **the promise of science**
for global good



Emily Caron

ecaron@rti.org

Jerry Copperthwaite

gcopperthwaite@rti.org

Rhymney Weidner

rweidner@rti.org