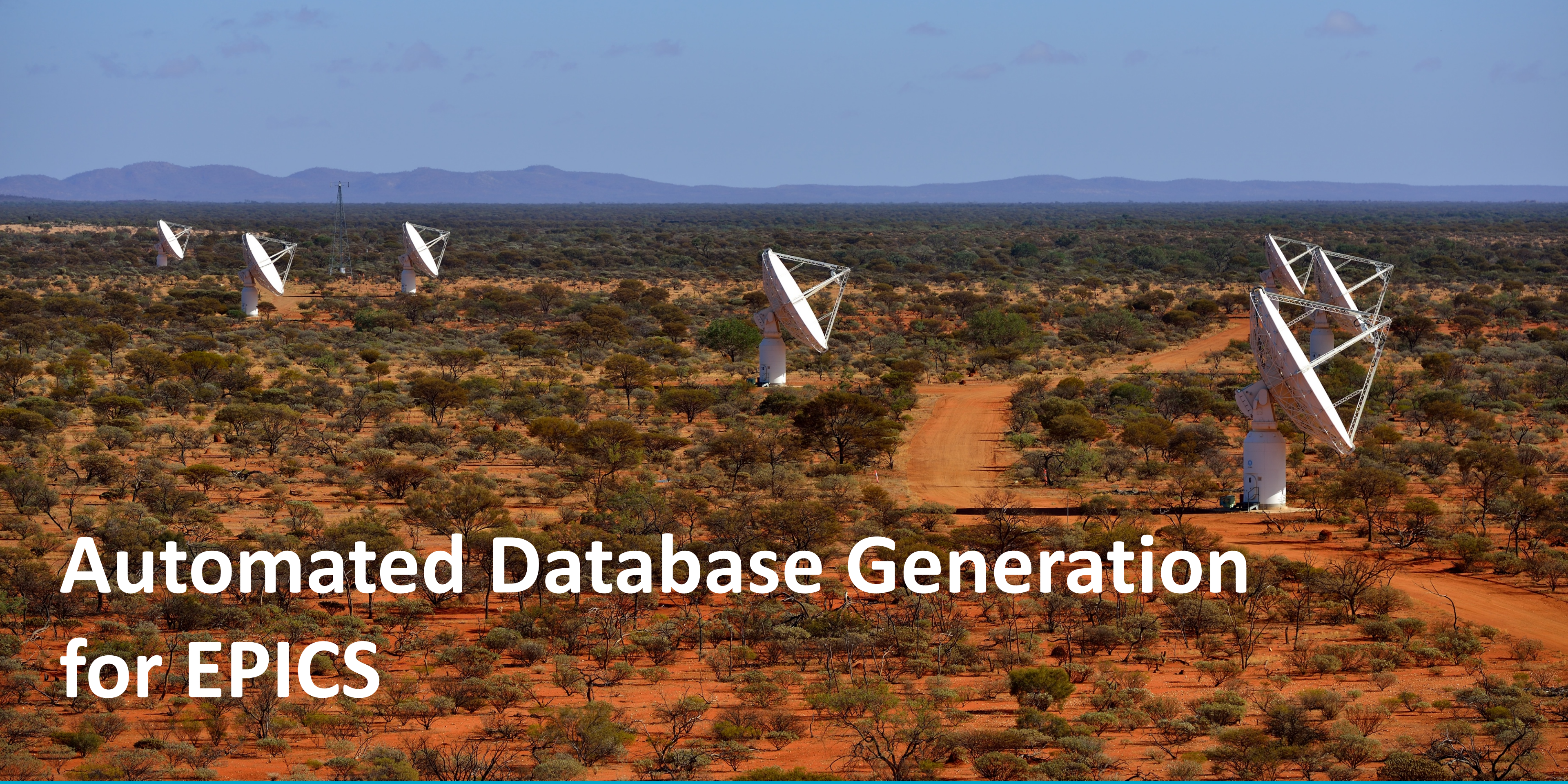


Automated Database Generation for EPICS

Zoe Taylor – CSIRO Astronomy and Space Science

November 2018

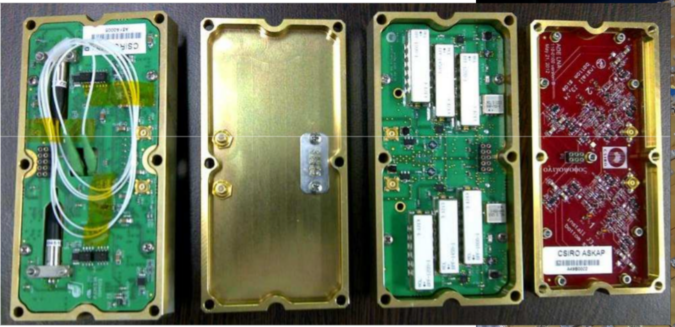
EPICS Collaboration Meeting, Melbourne

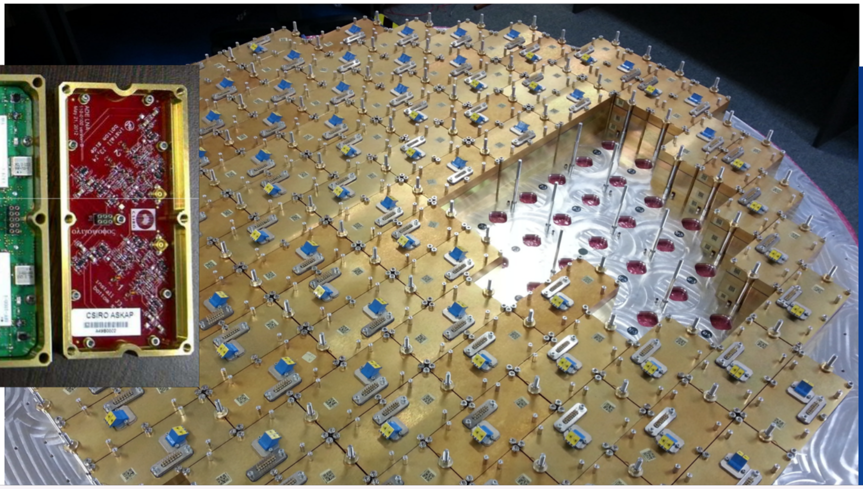
ASTRONOMY & SPACE SCIENCE

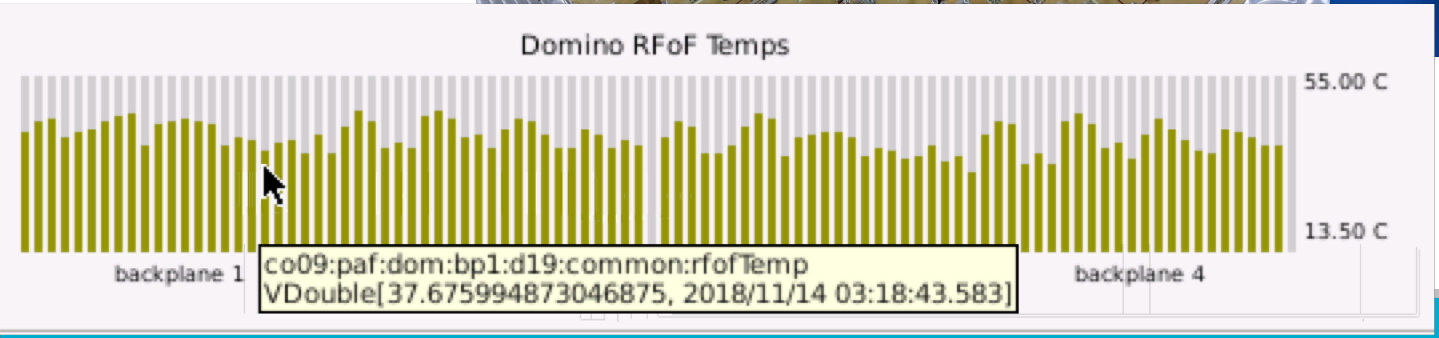
www.csiro.au

Complex subsystems - The receiver

- The Phase Array Feed is composed of 96 “dominos” in a checkerboard array
- Each domino has three sub components – RFoF (Radio Frequency Over Fibre), Filter and LNA (Low Noise Amplifier)







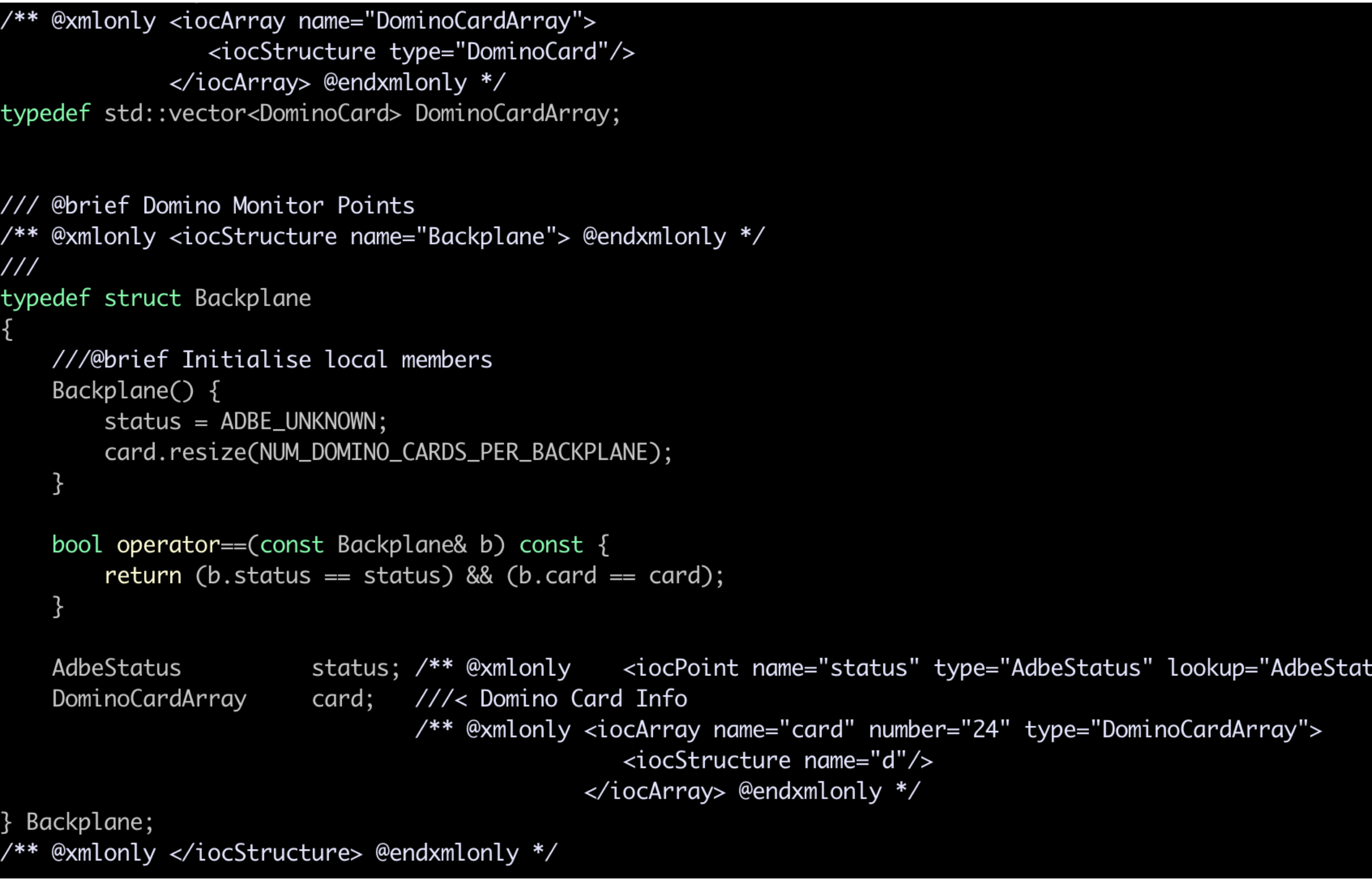
```
co09:paf:dom:bp1:d19:common:rfofTemp
VDouble[37.675994873046875, 2018/11/14 03:18:43.583]
```

AdbeParser

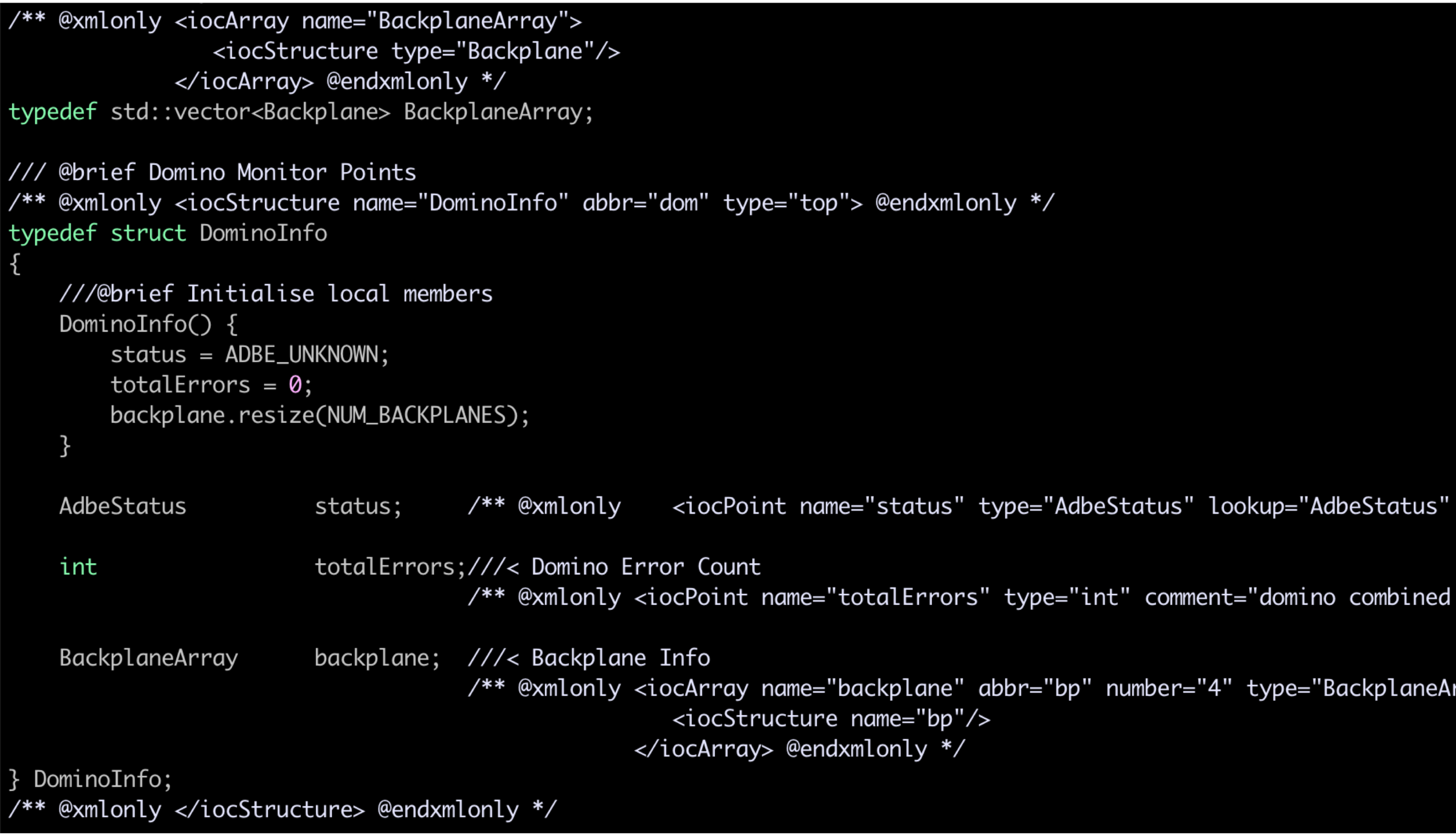
- A python tool for generating EPICS records from XML embedded in code
- Also generates
  - Datapoints for our in-house database archiving tool, MoniCA.
  - XML for CSS BOY GUI displays (e.g. PV Tables)
  - XML for OSL scripts
  - C++ structure to asyn update code
  - Asyn parameter lists

XML Tag	Usage
iocPoint	defining an EPICS record
iocEnum	defining an enumeration
iocEnumValue	defining an enumeration value
iocFunction	defining an IOC Function
iocStructure	grouping a set of iocPoints
iocArray	vector – for duplication of points

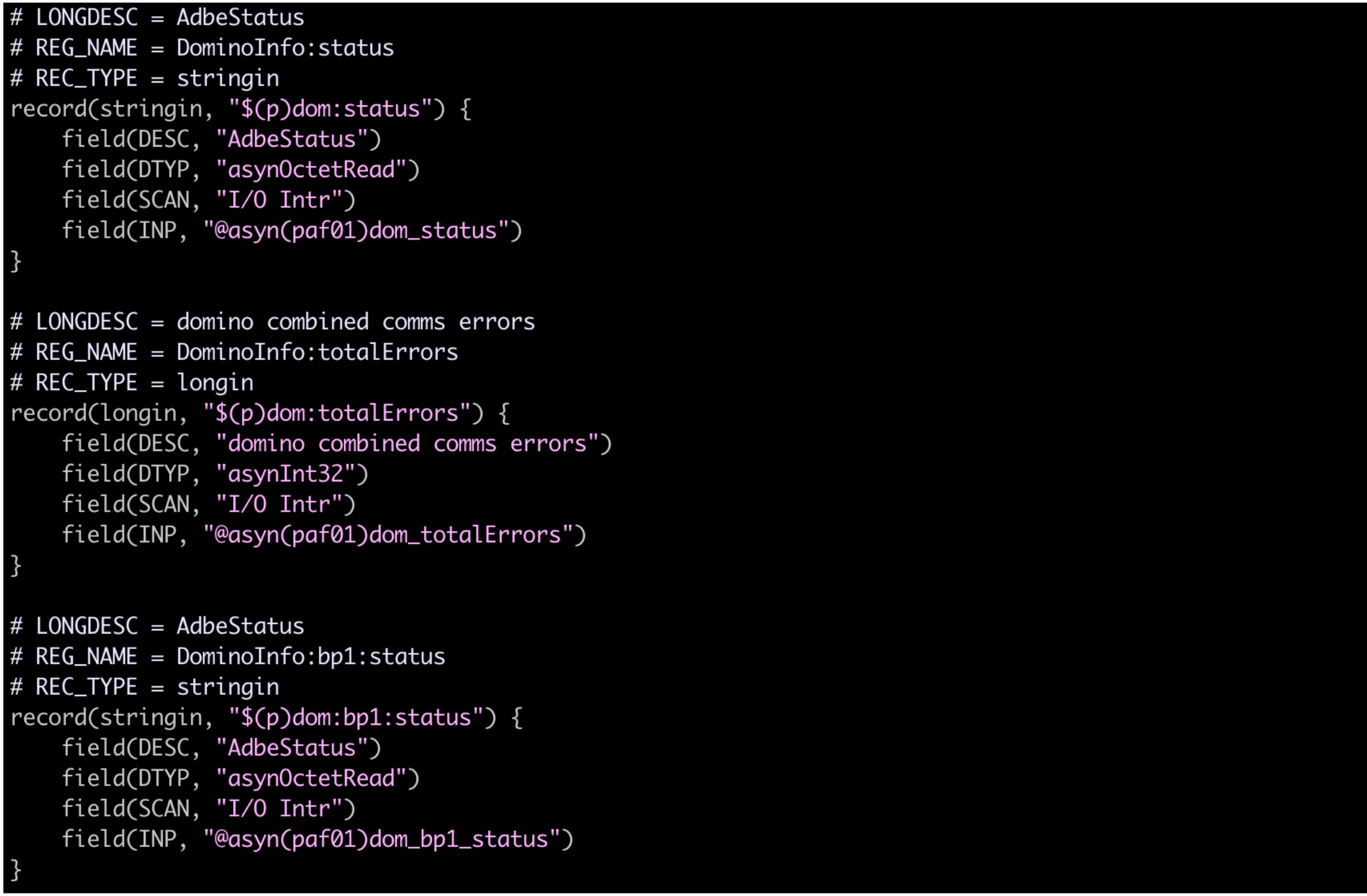
```
/** @xmlonly <iocStructure name="DominoCard" type="expose"> @endxmlonly */
///
typedef struct DominoCard
{
    ///@brief Initialise local members
    DominoCard() {
        status = ADBE_UNKNOWN;
        chanCtrl.resize(NUM_CHANNELS_PER_DOMINO);
        chanMon.resize(NUM_CHANNELS_PER_DOMINO);
    }

    bool operator==(const DominoCard& b) const {
        return ((b.status == status) &&
                (b.common == common) &&
                (b.chanCtrl == chanCtrl) &&
                (b.chanMon == chanMon));
    }

    AdbeStatus          status;     /** @xmlonly    <iocPoint name="status" type="AdbeStatus" lookup="AdbeStatus" comm
    DominoCommonInfo    common;     ///< Domino Common Info
                                    /** @xmlonly    <iocStructure name="common"  type="DominoCommonInfo"></iocStructur

    DominoChanCtrlArray chanCtrl;   ///< Domino Channel Control Info
                                    /** @xmlonly    <iocArray name="chanCtrl" number="2" type="DominoChanCtrlArray">
                                                        <iocStructure name="ctrlCh"/>
                                                    </iocArray> @endxmlonly */

    DominoChanMonArray  chanMon;    ///< Domino Channel Monitoring Info
                                    /** @xmlonly    <iocArray name="chanMon" number="2" type="DominoChanMonArray">
                                                        <iocStructure name="monCh"/>
                                                    </iocArray> @endxmlonly */
} DominoCard;
/** @xmlonly </iocStructure> @endxmlonly */
```



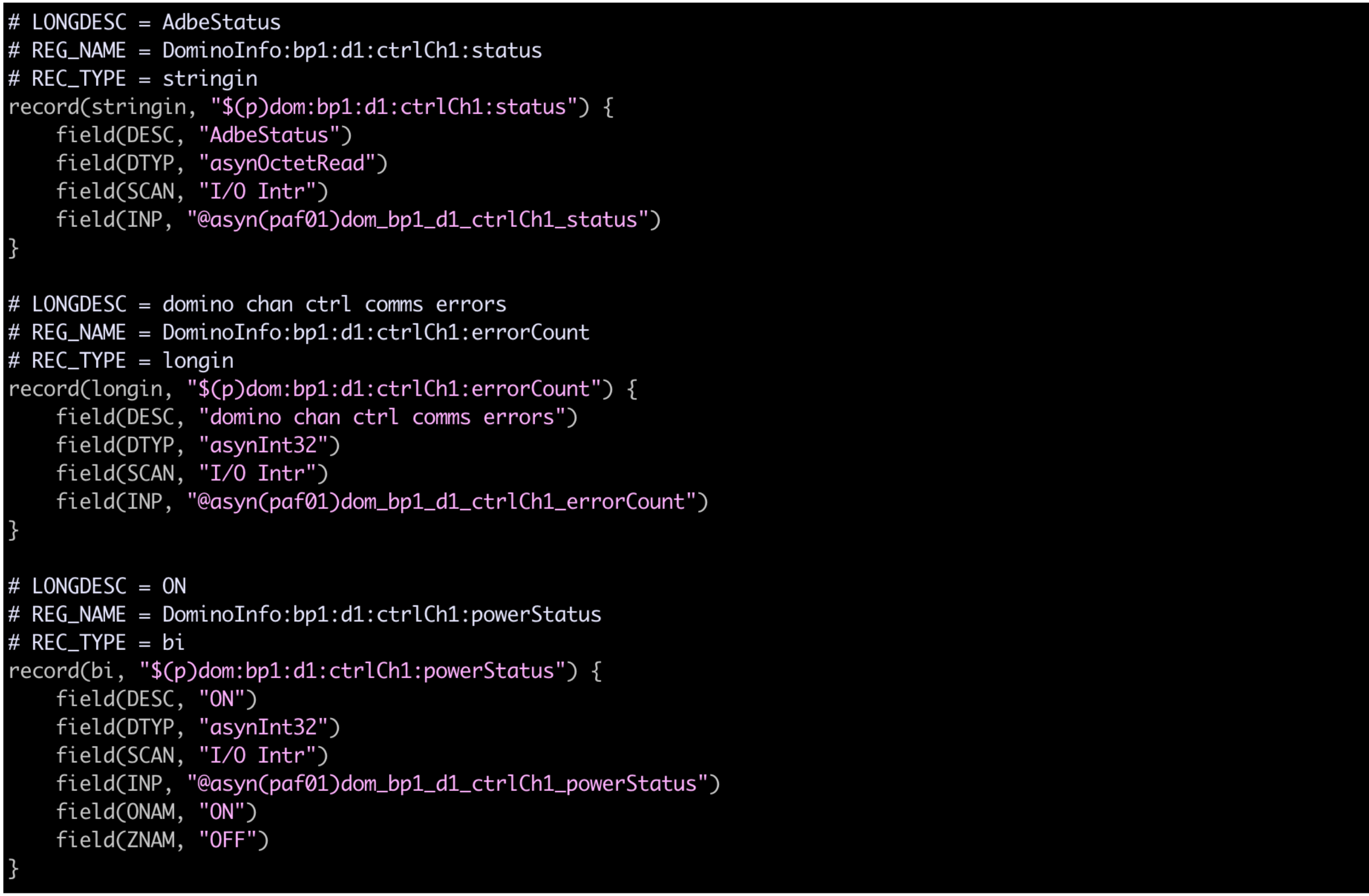
```
/** @xmlonly <iocArray name="DominoCardArray">
                <iocStructure type="DominoCard"/>
             </iocArray> @endxmlonly */
typedef std::vector<DominoCard> DominoCardArray;


/// @brief Domino Monitor Points
/** @xmlonly <iocStructure name="Backplane"> @endxmlonly */
///
typedef struct Backplane
{
    ///@brief Initialise local members
    Backplane() {
        status = ADBE_UNKNOWN;
        card.resize(NUM_DOMINO_CARDS_PER_BACKPLANE);
    }

    bool operator==(const Backplane& b) const {
        return (b.status == status) && (b.card == card);
    }

    AdbeStatus          status; /** @xmlonly    <iocPoint name="status" type="AdbeStatus" lookup="AdbeStat
    DominoCardArray     card;   ///< Domino Card Info
                                /** @xmlonly <iocArray name="card" number="24" type="DominoCardArray">
                                                <iocStructure name="d"/>
                                             </iocArray> @endxmlonly */
} Backplane;
/** @xmlonly </iocStructure> @endxmlonly */
```



```
/** @xmlonly <iocArray name="BackplaneArray">
                <iocStructure type="Backplane"/>
             </iocArray> @endxmlonly */
typedef std::vector<Backplane> BackplaneArray;

/// @brief Domino Monitor Points
/** @xmlonly <iocStructure name="DominoInfo" abbr="dom" type="top"> @endxmlonly */
typedef struct DominoInfo
{
    ///@brief Initialise local members
    DominoInfo() {
        status = ADBE_UNKNOWN;
        totalErrors = 0;
        backplane.resize(NUM_BACKPLANES);
    }

    AdbeStatus          status;     /** @xmlonly    <iocPoint name="status" type="AdbeStatus" lookup="AdbeStatus"

    int                 totalErrors;///< Domino Error Count
                                    /** @xmlonly <iocPoint name="totalErrors" type="int" comment="domino combined

    BackplaneArray      backplane;  ///< Backplane Info
                                    /** @xmlonly <iocArray name="backplane" abbr="bp" number="4" type="BackplaneAr
                                                    <iocStructure name="bp"/>
                                                 </iocArray> @endxmlonly */
} DominoInfo;
/** @xmlonly </iocStructure> @endxmlonly */
```

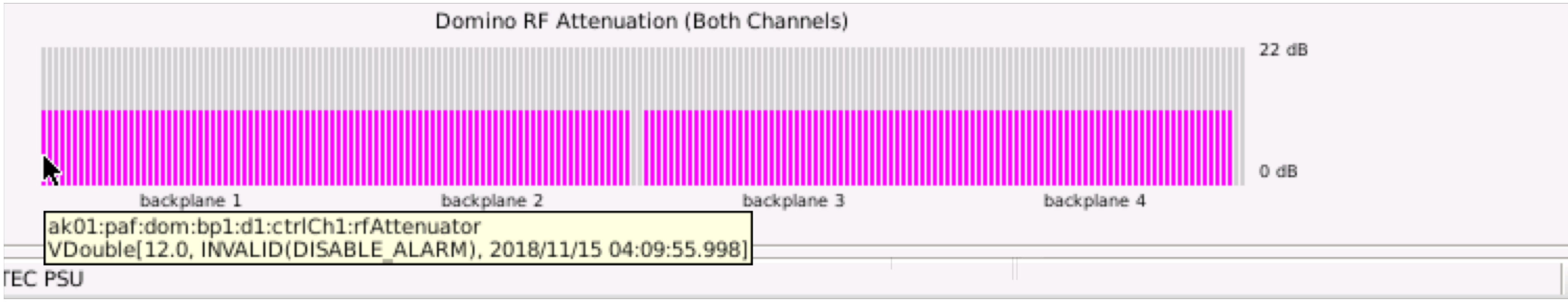


```
# LONGDESC = AdbeStatus
# REG_NAME = DominoInfo:status
# REC_TYPE = stringin
record(stringin, "$(p)dom:status") {
    field(DESC, "AdbeStatus")
    field(DTYP, "asynOctetRead")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_status")
}

# LONGDESC = domino combined comms errors
# REG_NAME = DominoInfo:totalErrors
# REC_TYPE = longin
record(longin, "$(p)dom:totalErrors") {
    field(DESC, "domino combined comms errors")
    field(DTYP, "asynInt32")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_totalErrors")
}

# LONGDESC = AdbeStatus
# REG_NAME = DominoInfo:bp1:status
# REC_TYPE = stringin
record(stringin, "$(p)dom:bp1:status") {
    field(DESC, "AdbeStatus")
    field(DTYP, "asynOctetRead")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_bp1_status")
}
```



```
# LONGDESC = AdbeStatus
# REG_NAME = DominoInfo:bp1:d1:ctrlCh1:status
# REC_TYPE = stringin
record(stringin, "$(p)dom:bp1:d1:ctrlCh1:status") {
    field(DESC, "AdbeStatus")
    field(DTYP, "asynOctetRead")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_bp1_d1_ctrlCh1_status")
}

# LONGDESC = domino chan ctrl comms errors
# REG_NAME = DominoInfo:bp1:d1:ctrlCh1:errorCount
# REC_TYPE = longin
record(longin, "$(p)dom:bp1:d1:ctrlCh1:errorCount") {
    field(DESC, "domino chan ctrl comms errors")
    field(DTYP, "asynInt32")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_bp1_d1_ctrlCh1_errorCount")
}

# LONGDESC = ON
# REG_NAME = DominoInfo:bp1:d1:ctrlCh1:powerStatus
# REC_TYPE = bi
record(bi, "$(p)dom:bp1:d1:ctrlCh1:powerStatus") {
    field(DESC, "ON")
    field(DTYP, "asynInt32")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_bp1_d1_ctrlCh1_powerStatus")
    field(ONAM, "ON")
    field(ZNAM, "OFF")
}
```

```
# LONGDESC = RF attenuator
# REG_NAME = DominoInfo:bp1:d1:ctrlCh1:rfAttenuator
# REC_TYPE = longin
record(longin, "$(p)dom:bp1:d1:ctrlCh1:rfAttenuator") {
    field(DESC, "RF attenuator")
    field(DTYP, "asynInt32")
    field(SCAN, "I/O Intr")
    field(INP, "@asyn(paf01)dom_bp1_d1_ctrlCh1_rfAttenuator")
    field(EGU, "dB")
    field(LOPR, "0")
    field(HOPR, "22")
    field(HIHI, "20")
    field(LOLO, "0")
    field(HHSV, "MAJOR")
    field(LLSV, "MAJOR")
    info(autosaveFields, "HIHI HIGH LOLO LOW HOPR LOPR HHSV LLSV HSV LSV HYST")
}
```



```
ak01:paf:dom:bp1:d1:ctrlCh1:rfAttenuator
VDouble[12.0, INVALID(DISABLE_ALARM), 2018/11/15 04:09:55.998]
```

Thank you

Astronomy And Space Science
Zoe Taylor

t +61 8 6436 8557
e Zoe.Taylor@csiro.au